# A Transport Layer Approach to Host Mobility

A Dissertation

Presented to

The Faculty of the Department of Computer Science

by

**Luiz Claudio Schara Magalhães**

In Partial Fulfillment

Of the Requirements for the Degree of

Doctor of Philosophy in Computer Science

College of Engineering

University of Illinois at Urbana Champaign

September 2005

# A Transport Layer Approach to Host Mobility

# Abstract

Support for host mobility is traditionally implemented in the link or network layer. In comparison, the transport layer has access to end-to-end information not available at lower layers. Coupled with inverse multiplexing techniques and error classification algorithms, such information can be used to achieve a significant improvement in data throughput to mobile hosts. This research centers on the design of a transport layer host mobility architecture. The main components of this architecture are a link layer manager, to mediate access to the communication infrastructure; a multiplexing transport protocol framework that allows the construction of multiplexing transport protocols suited to different classes of traffic; and finally, a location service that tracks the current address of the mobile host. We validate the design of our architecture through implementation of a suite of link-layer aware, multiplexing transport protocols, and use simulation to investigate aspects of congestion control and loss discrimination.

*To Father and Son, Memory and Hope*

# Acknowledgements

A Thesis is the result of the work of one. But it would not be possible without the help of many. Their love, companionship, support, advice and knowledge are intertwined in all these pages. This list of all who helped is far from complete.

First I would like to thank my family, my wife for having come to another country, leaving behind friends and family, and putting up with me through all the tense moments of a PhD; my mother for her constant support, shielding me from hard problems that could have distracted me from my way; my son, for showing me a greater meaning to life than I thought possible; and my extended family, the Schara and the Magalhães, for their love and quiet pride.

I would like to thank my advisor, Robin Kravets, for showing me how to cope with adversity; and Prof. Roy Campbell, for supporting me when I was looking for a Thesis. I would like to thank my friends Albert Harris, his wife Erin, Celso Mendes and his wife Marcia, for making Urbana seem like a second home. I would like to thank also Chandana Pairla and Fabio Kon, and all the other students in Roy's and Robin's group for their friendship and for helping me whenever I needed.

This work would not have been possible without the support from the Telecommunications Department at Universidade Federal Fluminense, which made do with one less professor during all the time I was away, and the financial support from the

Brazilian Government through CNPq, which supported me during four years.

I would like to thank my Master Thesis advisor, friend and colleague, Michael Stanton, for leading my way in the world of Computer Networks, when the Internet was a word known only to some initiates, and the Pontificia Universidade Católica do Rio de Janeiro, for preparing me for the challenges ahead.

Finally, I would like to thank Colégio de São Bento, for opening my mind to the world of science, which ultimately led me to be a Professor and a Scientist, and to the Class of 83, more than friends, brothers for life.

# Table of Contents

*“Eppur si muove”*
*Galileo*

# Chapter 1 Mobility in the Transport Layer

## *1.1 Introduction*

Wireless technology has come a long way since Guglielmo Marconi sent the first radio messages. The great triumph of wireless was connecting the whole world with geosynchronous satellites. However, the advent of fiber optics, with lower error rates and smaller delays than radio communication, made long-distance communication rely less and less on microwave links and satellite communication. A new renaissance of radio technology came with short distance communication, following the same evolution pattern of wired communication. First, cellular phones carried voice to mobile terminals. Then the same analog cellular technology was used to carry data to mobile computers. Finally, new wireless technologies were developed specifically for the transmission of data to mobile hosts.

Due to design trade-offs in power and range, the coexistence of many wireless technologies is expected. Software radios ([TU004], [DI003], [HA002], [RE002], [MI000]) may be used to allow a single device to mimic the multitude of radios from short-range (e.g. Bluetooth [BH001]) to mid (WiFi [IE099], [IE199], [IE299] [IE003], WiMax [IE004]) and long range (satellite communication). However, multiple, overlapping infrastructures to support different radio technologies are being installed, and should become ubiquitous in the near future. A host that uses wireless technology can be

mobile. Multiple overlapping wireless technologies create a connectivity rich environment, and a challenge: to allow mobile hosts take advantage of the various wireless technologies for improving its communication channel to the network.

The objective of this Thesis is to create a mobility architecture designed for wireless, connectivity rich environments. These environments pose an additional challenge: how to deal with the higher error rates of wireless channels as compared to wired technologies, because higher error rates diminish the throughput of the current transport protocols. Mobility also creates problems for transport protocols, because it invalidates measurements such as round trip time (RTT) and latency of the network path when the mobile changes its network attachment point. Besides that, communication may stop when changing from one network attachment point to another, or when the mobile enters an area with no wireless coverage. However, if the components of the architecture take the higher error rate in consideration, and are designed to use multiple wireless technologies at the same time, they can create a resilient communication channel that overcomes the problems presented by mobility and by wireless.

We first address the presence of multiple wireless technologies. To be able to use all the available wireless technologies, a mobile host has to connect to multiple networks simultaneously. This type of connectivity is called multi-homing. Normally, only specialized machines, like routers, are multi-homed, because of the low cost/benefit ratio of providing multiple wirings to generic hosts. However, the presence of multiple overlapping wireless infrastructures allows mobile hosts to be connected simultaneously

to more than one network, with possible gains in bandwidth and resiliency of communication. We propose a mobility solution based on dynamic multi-homing, where a host is connected to a constantly changing set of networks. The architecture of dynamic multi-homing needs three elements: a way of finding available wireless networks (and acquiring an address in each available network), transport protocols that use the multiple available networks simultaneously (with support for techniques such as inverse multiplexing and packet re-sequencing) and a location service to allow corresponding hosts to communicate with mobile hosts in their new network. Dynamic multi-homing has no parallel in stationary hosts[1], although dynamically acquiring an IP address and multi-homing are both well-understood techniques. Multi-homing allows data to be sent through multiple interfaces simultaneously, and poses the third challenge: how to use more than one interface for a single flow.

The main challenge for sending a single flow through different paths is that the Internet Protocol (IP), which implements the network layer in the Internet architecture, does not understand flows[2], and so cannot help with the distribution of a flow through multiple interfaces. Each packet is routed independently, and although it is possible to use multi-path routing algorithms to distribute a flow through multiple paths, these algorithms will not have end-to-end information such as loss rate, throughput and latency unless support

[1]Even if a wired host is multi-homed, it will rarely change its attachment points to the network. A mobile host will change attachment points because it is moving.

[2] Although IPv6 has a flow label in its header, its use has not yet been established.

for this is built into the network. This information is needed to choose intelligently how to use each path accessed by the different interfaces. In contrast, transport protocols have access to end-to-end information. If transport protocols are used for load-balancing flows through multiple interfaces, they can use end-to-end information to decide which interface should be used for each packet. Transport protocols can also deal with issues such as re-sequencing, because the different paths used for packets will cause packets that should be in sequence to arrive out-of-order at the receiver. Thus, our mobility architecture is implemented using transport protocols that are link-layer aware, which means that the transport protocols have information of which link-layers are available and the end-to-end characteristics of each path that goes through that link-layer.

Finally, there is the challenge of mobility. Mobility, which is the ability of a host to change its spatial location, is still a hard problem today. Hardware support for mobility is a fast-developing area, where many new standards are being proposed ([IE003], [IE004]). There are no known general patterns for device mobility yet, and there are very few applications developed specifically for mobility. On the other hand, there is a general consensus that mobility is desirable, and multiple infrastructures that support a limited form of mobility are being deployed. However, current mobility solutions are limited. Mobility solutions based on switching from one base station to another, that is, link layer mobility solutions, are tied to a single technology[3]. This leads to problems such as gaps in

[3] Cellular telephones can switch from one technology to another (e.g. analog to digital), but this is closely related to the hardware, and not yet a general solution, allowing the switch from any to any technology.

coverage, and does not allow users to switch from one technology to another without dropping their current connections. Mobile IP [PE002], a network layer solution, addresses this problem, but requires the deployment of another infrastructure (of home agents and foreign agents) to work. It may also be wasteful of network resources, because of the extensive use of tunneling. Additionally, Mobile IP is limited to using a single technology at a time, and the timing artifacts caused by handoffs and by changes in network characteristics negatively impact TCP [BL002].

Dynamic multi-homing solves the problem of mobility by changing the way the problem is currently dealt with. Current solutions, be they link-layer or network layer, strive to keep one connection to the network. When the current connection becomes unavailable, another connection is made. The mobile moves by switching from one connection to another. In contrast, a dynamically multi-homed mobile host has multiple network connections that change with time. This change may be caused by a failure of a base-station, or it may be caused by the mobile host moving away from base station range. If the environment is connectivity rich, at any given time the host is connected to multiple base-stations. The addition or deletion of a base station from the list of available base stations (of different technologies) does not affect a multi-homed host in a significant manner, because there are no traumatic breaks such as we see in current mobility solutions. Packets that are lost, because they were in transit when an interface became unavailable do not have to be treated in a special manner (although because the transport protocols built for multi-homing are mobility-aware, these lost packets do not trigger

congestion control measures). They are resent through the remaining interfaces.

The center of this work is the design of transport protocols for mobility and wireless communication technologies. In general, the wireless environment is not very agreeable to TCP, which took advantage of the small packet error rates present in today's wired networks to assume that losses are mostly caused by congestion. If that is coupled with blackouts (when the mobile host is in one area devoid of connectivity) and with the changes in path characteristics caused by the different attachment points of different technologies (which lead to packets following different paths in the network), the result is dismal performance, has lead to much research on how to improve TCP for wireless ([XY001] [ST198] [BA095] [BA096] [BA995] [BI099] [CA095] [BR096] [VA099]) and how to minimize handoff delay ([MA005] [BE104] [CA002] [HS003] [MA005] [RE003]). TCP does not have the bandwidth measurement tools that are needed for load-balancing a flow through multiple interfaces. Instead of retrofitting TCP, we decided to build new transport protocols designed for the connectivity-rich wireless environment.

The design of these new transport protocols meets a number of requirements. The new transport protocols use the IP infrastructure unchanged, and so, leverage the existing network. Most of the work is done at the edges of the network, at the mobile node and its peer, following the end-to-end principle [SA084]. The protocols take advantage of the existence of overlapping communication technologies, making the mobile host multi-homed by default. Traditional transport protocols have access to end-to-end information, such as path bandwidth, latency and error characteristics. In this design, this knowledge is

augmented by access to local information (the protocols are link-layer aware), so the protocols can react to the changes in path characteristics brought by mobility. The transport protocols can also take advantage of the overlapping wireless communication infrastructure to increase the bandwidth available to a single connection to/from the mobile host. To achieve this, a new congestion control algorithm was developed. This algorithm uses methods developed for bandwidth estimation. The better correlation of packet dispersal to available bandwidth given by a rate-based transmission algorithm can be used as input on how to divide a communication flow into different wireless technologies in a load balancing way, according to the available bandwidth in each end-to-end path given by accessing the larger network through that link layer.

The main contribution of this work is the paradigm change: moving mobility support to the transport layer. The key benefits of this solution are reduced handoff delay, increased bandwidth and reduced effect of losses. This approach provides seamless mobility support through the simultaneous use of multiple overlapping technologies. As an added benefit, this solution can take advantage of the multiple local base stations of different technologies, enabling bandwidth aggregation. Using inverse multiplexing, the transport layer enables a single application-level flow to be transmitted through multiple link layers, with gain in bandwidth, robustness and handoff delay.

The complete mobility solution is composed of three parts. The first is the Link Layer Manager (LLM), which is an entity responsible for link layer discovery and IP layer configuration. The second is a suite of transport protocols designed with the above

requirements and adapted for the traffic they will be carrying. The third is a location service to allow mobile hosts to be contacted when away from their home network.

This work is centered on mobility-aware transport protocols. Due to the problems TCP has with wireless environments given their higher error rate, and the decision to create protocols that could use more than one link layer simultaneously, the new protocols were designed from the ground up. Most of the contributions of the Thesis are related to the transport protocols, including the new congestion control mechanism and a heuristic to distinguish if a loss was caused by congestion or by transmission error, using information available at the receiver.

The contributions of this Thesis are:

a) The creation of new mobility architecture based on multiplexing, mobility aware transport protocols [MA101]. The architecture is presented on Chapter 2.
b) The study of a new congestion control algorithm. The Homeostatic Congestion Controller is presented on Chapter 3.
c) The study on how to discern transmission losses from congestion losses [MA003], using the mechanisms given by the Homeostatic Congestion Controller. The heuristic for loss discrimination is presented on Chapter 4.
d) The design, implementation and evaluation of a multiplexing protocol for multimedia data. Chapter 5 describes the Multimedia Multiplexing Transport Protocol (MMTP [MA001]), which uses the Homeostatic Congestion Controller and loss discrimination presented on the previous Chapters, and adds the

characteristics needed for multimedia data, such as partial reliability, application framing and priorities.

e) The design, implementation and evaluation of a multiplexing protocol for bulk data. In Chapter 6 the Reliable Multiplexing Transport Protocol (RMTP [MA201]) is presented. It uses the same mechanisms as MMTP, but has full reliability and sequencing.

The next Section describes the challenges of introducing mobility in the Internet, due to coupling of addressing with routing, which explains the inefficiencies that saddle Mobile IP. It shows the advantages of an architecture that can use the IP infrastructure unchanged and still achieve mobility.

## *1.2 Mobility and Internet Addressing*

In this section, we will explain how the early design decisions that tie IP addresses to routing make mobility a difficult problem. We will show the limitations of link-layer based mobility architectures, and also how Mobile IP is hampered by not being able to use end-to-end information available to transport layer entities.

Internet addressing follows a two-tier model. The upper tier is the host name, the human-readable part of the Internet addressing scheme. When a user types an URL in a browser window, the host part of the URL is translated by using a database called the DNS (Domain Name System [MO087] [MO187]) into an IP address, which is the lower tier.

The IP address will then be used for communication with the desired host.

This two-tier model works very well, because the added level of indirection allows for flexibility in naming. Hosts may have multiple names, or a name may point to several hosts. It also frees the names from any connection to locality. While the DNS database was designed with a naming scheme that clusters names according to their characteristics[4], this is not enforced. Any company can have a ".com" name, even those without an US presence, and the small country of Tuvalu has a valuable Internet location because of its ".tv" ending, making it desirable for TV stations all around the globe. Even if the naming scheme was enforced, large companies such as IBM could have a single name termination (domain) comprising offices scattered all around the globe. Although logically those machines all belong to IBM, they may be geographically, and even topologically (network-wise) very far apart.

In contrast with DNS names, IP addresses have a strong sense of locality, especially after the near collapse of routing tables at the central routers led to the creation of CIDR [LI093] [FU093], the Classless Inter-Domain Routing. Today, the world is divided into regions with very well defined IP number-range associated with them, although Internet (network) topology does not map directly to physical topology. The reason there is a close association of IP numbers with the physical topology of the Internet is due to the

[4] That is, education institutions have a name terminated with "edu", commercial institutions "com", institutions belonging to countries outside of the US their 2 letter ISO code.

way the host IP number is used in Internet routing. For large scale routing[5], IP addresses are divided in two parts, a network part and a host part. The host part is important only at the last step of the way, when the IP datagram is delivered to its final destination, but the network part is used to index the next hop table at each of the routers between source and destination.

This design is very efficient, and is probably one of the reasons why IP technology has been so successful, driving all its competitors (IPX/SPX [TA003], OSI [RO090]) out of the market. However, this design does not allow for mobility, because a host IP number is firmly attached to the network. If a host moves from one network to another, it has to change its IP number. Using the old IP number from its home network, the machine may still be able to send packets (though today most border routers will filter out this packets as topologically incorrect), but it will not receive any packets addressed to it. Those packets will be routed to the home network where the machine no longer is.

It is important to make a distinction between nomadic and cellular mobility [ST002]. In nomadic mobility, the host is not attached to the network while it moves from one place to another. A student that uses a laptop at school and at home is an example of nomadic computing. When at school, the laptop uses a number belonging to the school's network. When at home, the laptop has to acquire a different IP number, belonging to the home network or to the Internet Provider the student has an account on. Because there is no

[5] Internal routing, or routing done inside an Administrative Domain may further subdivide the IP address, creating sub-nets [CO095].

need for seamless transitioning, as the laptop is not communicating while moving, the only infrastructure needed for this type of mobility is a DHCP [DR097] server at both locations (or a PPP [SI094] server in the case of a Internet Service Provider). The laptop uses DHCP to acquire a new IP address each time it is turned on in a new location.

In contrast, cellular mobility maintains connectivity while moving. There are two different ways to implement cellular connectivity today. The first, as the name implies, is to use some kind of wireless technology to maintain link layer connectivity while the host moves, exactly like a cellular phone. In general, although very distinct cellular technologies have been developed, all of them are based in the principle that the cellular phone or mobile host is capable of measuring the signal strength it is receiving from the base station it is currently connected to, and the signal strength of other base stations that are in its vicinity. When the signal strength from another base station is greater than the signal strength of the base station the host is currently connected to, the host switches to the new base station (this is called a hand-off). There has to be some hysteresis so there are no oscillations if the host is exactly at a boundary where signal strength from two or more sources has equal values, and the network has to deal with forwarding the packets to the right base station after the hand-off has occurred. This allows seamless connectivity, but ties the mobile host to a single technology, because the host can only switch between base stations of the same technology. The second way to implement cellular connectivity is to use a network layer approach, Mobile IP (MIP [PE002] [PE098] [SO097]).

Mobile IP allows hosts to move from one network to another while keeping its "home"[6] IP address by creating a tunnel (IPIP [PE096] GRE [FA000]) from the home network and the current network. All packets addressed to the host are sent normally to the home network following normal IP forwarding, where they are encapsulated by a server called the "home agent" and sent to the new network destination. At the new network, they are de-encapsulated by another server, called the "foreign agent" and delivered to the mobile host for normal IP processing. Because the last step does not use IP forwarding, but only ARP [PL082] or the tables maintained by the foreign agent (if the foreign agent knows the machine that is being addressed by the encapsulated packet is now residing in the same physical network the foreign agent is connected to), the packet can be sent directly from the "foreign agent" to the mobile host. The whole process is transparent to the host applications, but uses additional network resources because packets have to travel to the intermediate destination, regardless of the actual proximity between the mobile hosts and the communicating peer[7]. Mobile IP also requires additional infrastructure, because there has to be at least a server at the home network to forward the packets (a mobile host may potentially be its own foreign host, if it can acquire an IP address at the foreign network).

---

[6] The home IP address is the IP address acquired when the mobile host first connected to the network, during the current communication session. It may be even a temporary address given by a DHCP lease, although in this case this lease has to be refreshed by its home agent. A simpler case is when the home address is a fixed IP address.

[7] There are routing optimization to minimize this problem, see [CH002]

The reason why it is so important to keep the same IP address while moving is that the higher level (transport) protocol (TCP) uses the source and destination IP numbers as part of its connection identifier (the other part is composed of the source and destination ports). Should a machine change its IP number, all of its TCP connections would immediately be undone. To allow TCP to work unchanged while still allowing hosts to roam is one of the reasons why Mobile IP was designed as it is.

Mobile IP does allow a mobile host to switch from one communication technology to another. For instance, a student laptop may be connected to the campus wired network while working at the University library, switch to a GSM EDGE [ER003] network while moving from the library to the classroom and finally at the classroom switch to the campus wireless network, without for instance stopping an FTP [PO085] download. Although potentially the laptop has a different IP address at each network it connected to, all IP packets it received once away from its home network were encapsulated in another IP packet addressed with either its address on the foreign network or its foreign agent local address. This encapsulation [PE096] creates a tunnel from its old location to the new, and all packets destined to it that ended in its original network were tunneled to the mobile's new location. This freedom of using any available network begets a question: when should a mobile host transition from one network to another?

In a world where network connectivity is rare, to answer this question is not difficult. The mobile host should hand-off to a new network whenever the current network becomes unavailable. In a connectivity-rich world, on the other hand, other requirements may

come into play. For instance, the cost associated with the technology being used may be an important factor. Another factor in choosing one communication technology over another when more than one technology is available may be the quality associated with the connection, which may be related to factors such as the bandwidth available, the delay or even the jitter experienced by packets sent by that interface. If more than one application (or even more than one TCP connection in the same application) is actively using the network at one time, this may be not clear cut, as each destination may be better served by a different technology. In this case, which communication technology should be used?

These questions are not answered by Mobile IP. In fact, Mobile IP cannot answer these questions because, by being a network layer protocol, IP is not concerned with end-to-end measurements that could gather data to decide one way or another. By deciding on Mobile IP's transparency to transport layer protocols, all measurements that TCP makes, for instance, on the RTT experienced by a connection and available bandwidth on the path currently being used cannot be taken into account.

Another problem with measuring bandwidth is that networks are an example of Heisenberg's Uncertainty Principle [HE027] [FO097] [GR004]. When a TCP connection measures how much bandwidth it can use, it is in fact changing the rate of all other TCP connections (and connections of other reactive protocols) that are using segments of the

same path. It is impossible to measure available bandwidth[8] without actually using the network, and thereby changing its conditions. Therefore, if many communication technologies are available at a single location, choosing which is the best technology to use may require constant measurements, because of changing conditions in the access network, in the network path and in network destinations. In [BE004] a measurement called "qualities of the interfaces (QoI)" was proposed, together with a technique for constant measurement to choose the best interface.

The mobility architecture proposed in this Thesis answers these questions by designing transport protocols that use all network technologies that are available in the mobile host environment. The protocols actively measure the end-to-end performance of each connection, allowing data to be distributed into every available link layer according to the best possible usage.

[8] In here we mean end-to-end measurements, without help from active elements in the network. If a resource reservation scheme [BR097] is used throughout the network, then the network knows how much bandwidth is available, and an interested application may query it, only slightly changing network conditions by sending the query packet and receiving the response.

# Chapter 2 Mobility Architecture

## *2.1 Introduction*

This Chapter presents the transport layer based mobility architecture. A mobility architecture has two concerns: first, how a mobile host can communicate while moving; second, how can a corresponding host find a mobile host. In our architecture, we have a third concern, which is how to use in the best way possible the heterogeneous, overlapping wireless infrastructure mentioned in the last Chapter.

The architecture can be divided in two parts, the internal elements, which run on the end-systems, and the external elements, which run in the infrastructure. The internal elements are the transport protocols themselves, and the operating system module needed for managing multiple link layers. The protocols allow the mobile to communicate while moving, by being uncoupled from the underlying IP infrastructure. The operating system module allows mobility by finding and offering the use of different wireless (or wired) technologies to the transport protocols. The external elements are the constituents of the location service, which allows a corresponding host to communicate with a mobile independent of its location, by giving the corresponding host a pointer to the mobile's host current location.

The challenges are how to locate the wireless network infrastructure and acquire IP addresses for each technology found. Once the link layer is accessible, the challenge is to how to take advantage of the potential multi-homed characteristic of the mobile host, by

measuring the end-to-end characteristics (such as available bandwidth and delay) of the path through that link layer, so data can be sent according to the application's needs, and how to distribute the date into multiple link layers. Also, because wireless link layers are more prone to transmission errors, how to optimize the transport protocols so they will work well with higher error rates. Last, there is the challenge of tracking the location of a mobile host so data will be correctly routed to it. Although this architecture solves the problem of routing by using local IP addresses once a connection is established, to establish a connection it is necessary to find at least one of the set of IP addresses used by the mobile host.

Historically, mobility has not been an integral part of the Internet architecture. Mobile hosts as we know them today are a recent development. The original design of the Internet had mainframes as the primary constituents. Although subsequent versions of IP were able to change the focus from a few multi-user systems to a multitude of single user systems, the basic idea of a fixed infrastructure remained. This created an environment where it made sense to optimize the internetworking protocol for a structure that changed on a relatively large time-scale, leading to the current hierarchical addressing and routing architecture.

Adding mobile hosts to the Internet is further complicated by the absence of established usage patterns for mobile hosts. It took telephony more than one hundred years to go from landlines to wireless, although the basic service is well understood. Even cellular service can be seen as an extension of wireless phones, which existed long before public

service was offered. The same data on usage patterns is still not established for mobile hosts, and it is an interesting area of research [KO002]. Computer usage is still evolving, and mobile computers are a very recent addition. Until the needs of mobile users are well known, it will be hard to design an optimal architecture, as was done for wired static hosts.

The hierarchical, geographically related IP address space we have today is a compromise. The original tenets of address distributions did not consider any geographical location. Addresses were distributed on a first-come-first-served basis, and neighboring addresses were strewn across the world. The philosophy was that IP addresses should be free from geographical meaning in the same way DNS network addresses can span different IP network addresses. Unfortunately, the current router technology did not permit the multitude of non-related IP networks. The scattering of network addresses around the world meant that routing tables had to list every network: geographical supernetting compression was not possible. In an effort to keep router loading under control, IP address distribution became geographical [HI093].

Router technology is much better today. It is doubtful that supernetting would be indispensable with current technology, which does not mean that it is not welcome. However, this type of engineering decision is the same that hampers the deployment of mobile hosts. If a different routing scheme were used, e.g., if the Internet were bridged

instead of routed[9], there would be no challenge in moving attachment points. On the other hand, if there were a scalable solution that allowed single IP addresses to be routed independently, with changes in the routing structure on the time-scale of host mobility, there would be no need of special mobility solutions. The internetwork fabric itself would take care of mobility. IPv6 [DE098], facilitates certain aspects of mobility, because a mobile host is able to create its own care-of address using link-local address and automatic address configuration, by combining the advertised subnet prefix with own hardware address.

Our mobility architecture works well with IPv4 and IPv6, and can take advantage of the new facilities offered by IPv6. There are two central ideas in the design of the mobility architecture: transport protocols should be able to utilize all communication resources available in one area and transport protocols should be aware of mobility to regulate the flow into (possibly) multiple link layers.

It is possible to design an IP-only mobility solution that uses all communication resources available in one area, by assuming that mobile hosts are multi-homed by nature and design. Using multipath routing algorithms ([VU001], [LE005]), a single flow could be divided across the multiple interfaces. The problem with this solution is that current transport protocols do not expect constant reordering of packets, a side effect of using multiple routes with varying delays. A simple solution is to fix each flow to one interface.

[9] This is not to advocate a bridged global network. However, it is the optimization of routing using network address prefixes that makes host mobility a hard problem.

This approach does not allow a single flow to attain the maximum available bandwidth of the aggregated resources, and generates another problem: how many flows should be sent through each interface? A transport layer approach solves these problems naturally. Resequencing is done by buffering packets at the transport layer, which expects packets to be delivered out-of-order. Load balancing is attained by measuring the available bandwidth on each channel and dividing the flow according to the percentage of available bandwidth each channel represents. Because IP lacks the mechanisms for measuring bandwidth, and because TCP does not expect packets to be reordered, an optimal TPC/IP multipath mobile solution is hard to be constructed. Non-optimal solutions, on the other hand, require only the construction of a multipath router and flow classifier at the mobile host.

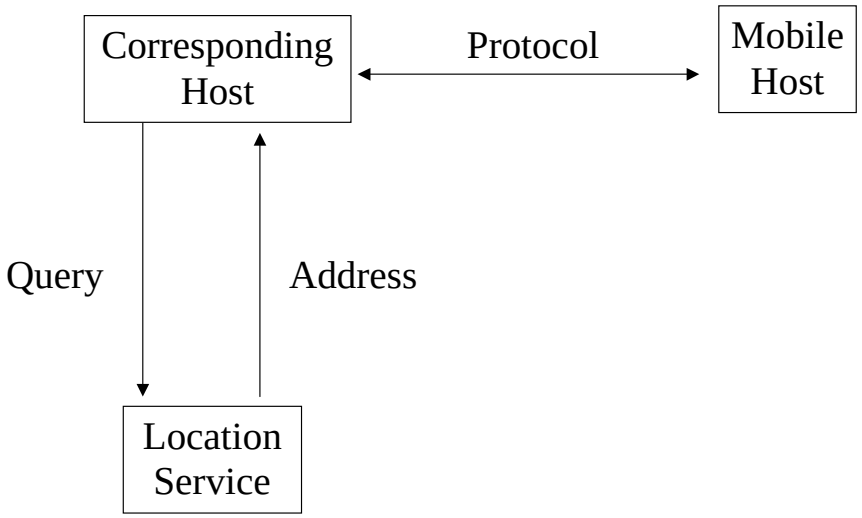



**Figure 1: External Elements of the Mobility Architecture**

The main contribution of this Chapter is the architecture for transport layer based end-to-

end mobility. It is composed of two parts, the infrastructure portion and the internal components. Under IPv4, an address allocation service and a location service are needed to serve the mobile with local addresses and to allow external users to access these local addresses. Under IPv6, these services change due to the facilities of creating non-conflicting local address by using globally unique hardware addresses, but the actions remain the same. The internal components are the Link Layer Manager and the suite of protocols. A representation of the external elements of the architecture can be seen in Figure 1. A Corresponding Host will query the location service for the current address of the Mobile Host, and then communicate directly to it using one of the mobility-aware transport protocols.

This Chapter is organized as follows. Section 2.2 presents the general architecture for the transport protocols. This architecture will be further examined in the following Chapters. Section 2.3 presents the Link Layer Manager, which is responsible for managing the multiple, dynamically varying link layer interfaces and its IP addresses. Section 2.4 presents the Location service, needed for finding the mobile host when it is away from home. Finally, Section 2.5 presents the conclusion and future research.

## 2.2 The Channel Framework

One of the goals of this research is to enable the simultaneous use of multiple link layers for a single data flow. To this end, we define a *channel*, which is an end-to-end, transport layer connection that encompasses all available link layers and multiplexes the data of a

single flow into these links. The sending application sees a single *channel*, one transport layer interface that remains stable. The transport layer protocol receives the data from the application and sends it through *sub-channels*, network layer sockets mapped to different link layers. At the other end, the transport layer gathers the data from the *sub-channels* and delivers it to the peer application. To create a *sub-channel*, the protocol needs information about what link layers are available. The transport layer has to be *link layer aware*, although it will not communicate directly to the link layer, relying instead on the abstractions offered by the network layer.

The number of *sub-channels* will depend on the availability of network access points for different technologies in the range of the mobile system. The addition and deletion of *sub-channels*, as the mobile enters or leaves the coverage area of one access point, is transparent to the application. The only side effect noticeable by applications of *sub-channel* addition and deletion is the variation in available bandwidth. It is possible that not all network interfaces in the mobile host will be used constantly. The user may impose restrictions on usage depending on cost, the operating system may limit usage depending on the levels of battery power, and large mismatches of link layer characteristics may prevent the use of all available interfaces simultaneously. On the other hand, it is difficult to estimate path characteristics without actually sending data through a path. Additionally, raw bandwidth is not a good indicator of available bandwidth, because the numbers of simultaneous users or the bit error rate experienced by the mobile may sharply limit the throughput of an interface.

The main benefit of multiplexing data into multiple interfaces is the added bandwidth on the last hop, the common bottleneck link for mobile hosts. There are many other benefits, such as providing a natural way to deal with host mobility, smoother handoffs and greater link reliability. An application does not have to be aware of host mobility. On the other hand, more sophisticated applications may benefit from extra information, so an interface to query the main *channel* characteristics is available, together with an asynchronous notification system to report changes in these characteristics.

### 2.2.1 Architecture

The main components of our *channel* architecture are depicted in Figure 2. There are five entities: the application, the transport layer, the network layer, the link layer and the link layer manager; and three interfaces: the application/transport layer interface, the transport layer/network layer interface, and the link layer notification interface. This work is centered on the transport layer. The *channel* is the end-to-end connection that can be accessed by the interface between the application and the transport layer. The transport layer receives data from the application and multiplexes this data into the available *sub-channels*. Each *sub-channel* is mapped to a link layer interface through the network layer. This is shown on the figure as the arrows that go from the transport layer to the link layer. The transport layer must be *link-layer aware*. Such awareness can be achieved through an entity that does link layer management (LLM in Figure 2).

As seen on the previous Section, the LLM is responsible for link layer discovery, IP address management, and for informing the transport protocol of the presence of new *sub-channels*. The LLM also informs the transport layer if a current network attachment is lost, which implies in the loss of the attached *sub-channel.*

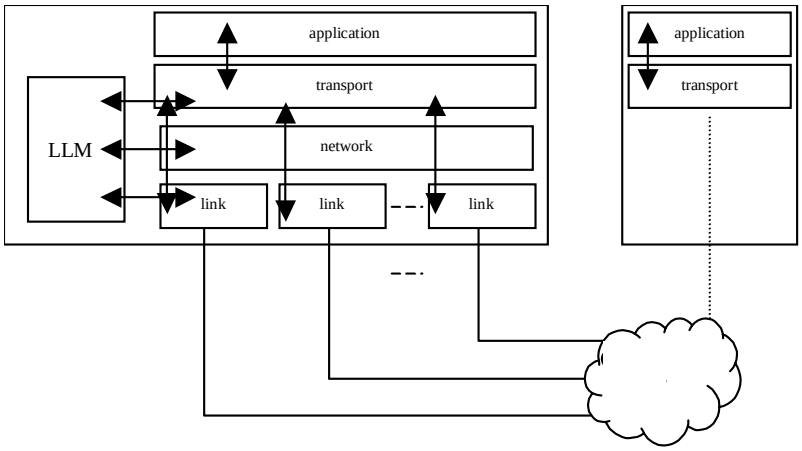



**Figure 2: The Channel Architecture**

### 2.2.2 The Application/Transport Layer Interface

For current applications, the most important interface is to the virtual *channel* offered by the transport layer. The interface to the *channel* mimics an interface to a non-multiplexed transport protocol. Thus, it can be used as such by an application. The application gains mobility and robustness, and need not be changed. A normal socket interface can be used. Signals are used for asynchronous notifications from the transport protocol to the application. A signal handler is installed by the application if it desires to be notified of events, generated for example if the transport protocol has to violate the parameters set by the application. Some of these parameters are defined below.

#### *2.2.2.1 Channel Parameters*

For a typical application in current use today, using the above-defined *channel* is an easy way to use bandwidth aggregation and mobility. The application opens a socket using one of the transport protocols that implements the *channel abstraction* and sends data normally. For applications that have special quality of service (QoS) needs and for controlling the amount of bandwidth each application gets, three parameters are defined: *max rate*, *latency*, *min rate.*

The *max rate* sets a hard limit on the amount of data sent/received. The application uses the *max rate* as a cap on the maximum amount of data it can send. For the transport layer, the *max rate* is used for flow control, and defines the maximum amount of data an application can receive in a unit time. A similar *max rate* is used at the network layer in a per-channel basis, which translates to the maximum amount of data the physical channel encapsulated by the link layer can sustain.

The *latency* is a measure of the time between when a packet is sent and received, or half a round trip time if the sub-channel is symmetric. Latency is generally transparent for applications. After steady state is achieved in a connection, the interarrival times are more important than the transit times. However, it is an important measure for the transport layer to decide if a new channel is added or not to protocol processing. Great mismatches in latency can increase the size of the buffers at the sender or receiver. If packets are sent in sequence, they will arrive out-of-order, which requires buffering at the receiver to reorder them. If ordering is done by changing the order packets are sent,

packets have to be buffered at the sender. The size of the buffers needed for ordering is directly proportional to the latency mismatch of the sub-channels.

The *min rate* is a QoS parameter ([NA099] [XU098]). An application can set the *min rate* to signal that the interarrival times of the packets in its stream are bound to that interval, and if they arrive later than that they will be useless. The transport protocol can use this knowledge to increase the number of packets that arrive in time if the aggregated bandwidth of the *channel* is smaller than the *min rate* by dropping packets at the sender.

By offering these parameters, the *channel* turns into a cross-layer communication middleware that frees the applications from having to implement their own timing algorithms. At the same time, it adds bandwidth aggregation, by the use of multiple channels, and mobility capacities, by the ability of adding and deleting channels.

### 2.2.3 The Sub-Channel Interface

The interface between the transport layer and network layer is also simple, with a small twist. Because generally all network access points will be connected to the Internet, any interface can be used to deliver any packet. On the other hand, it is necessary to gain control of which link layer is being used to transmit each packet. To that end, the protocols have to be *link layer aware*. Each packet that is sent is tagged with the appropriate interface, and will be sent through that interface regardless of the contents of

the mobile's routing table. This bypasses the IP routing layer. A way to achieve this has been developed for Linux systems, with the socket call SO_BIND_TO_DEVICE [ZA098].

### 2.2.4 Protocol Mechanisms

The protocols that implement the *channel* abstraction will all share the same basic mechanisms. The next Chapter will look into the congestion control mechanisms appropriated for mobile (wireless) hosts. The congestion control should be aware that not all packet losses in wireless environments are caused by congestion. The problem of discriminating transmission losses from congestion losses will be analyzed in Chapter 4, where a loss discrimination heuristic is developed. Mobility is a sub product of being multi-homed, and being able to change the set of available network connections. The mobility mechanisms will be discussed in Chapters 5 and 6, where protocol implementations are presented. The mechanism for inverse multiplexing will be examined below.

#### *2.2.4.1 Inverse Multiplexing*

Multiplexing is a technique used in telecommunications for using a single channel for multiple flows. The flows are intermingled at the sender (multiplexed) and separated at the receiver (demultiplexed). Multiplexing was done traditionally in the time domain, where the channel is given to each flow at certain times (Time Division Multiplexing TDM [HA005]) or in the frequency domain (Frequency Division Multiplexing FDM

[ST001]), where a channel (composed of a large frequency range) is divided into sub-channels, each with a part of the original frequency range. Today more advanced multiplexing techniques are used, such as Code Division Multiplexing [VI095], used in spread spectrum cellular phones.

In this work we want to send a single flow through multiple channels (with no redundancy). This is the reverse of multiplexing, therefore the name “inverse multiplexing”. There are other examples of inverse multiplexing, for instance, the technique of channel aggregation in ATM [CH098], and the simultaneous use of two ISDN channels through PPP multi-link [SK096]. These are link layer techniques, and require that the links being aggregated have the same end points, that is, they are a one-hop only technique, connecting one sender to one receiver. The link layers also have very well known capacity, and very similar delays, so it is straight forward to divide the flow into the sub-channels.

In contrast, sending a flow through different wireless technologies poses a series of challenges. To know the amount of data that can be sent it is necessary to measure the available bandwidth in the path from sender to receiver. The challenge is to measure available bandwidth. Packets sent through channels with different delays will arrive at different times. Therefore, out of order arrival should be common. The challenge is to reorder the packets. Finally, the amount of buffer space needed at the sender and receiver for performance, reliability and reordering depends on the delay associated with each channel. The challenge is to measure delay and allocate the right amount of buffer space.

The rate-based transmission mechanism and the Homeostatic Congestion Controller result in the knowledge of the available bandwidth in each channel. The delay on the path can be measured at the beginning of the connection and at the addition of each new interface by measuring the round trip time of the initial packet exchange. Because buffer space is normally allocated at the beginning of a connection, it is possible that the addition (and deletion) of interfaces will result in under-optimal buffer space (where the throughput will be limited by the number of packets that can be in flight). This can prevent interfaces from being used, because their bandwidth/delay product is too high.

The mechanism used for inverse multiplexing is basically to have a single output queue being serviced by the multiple sub-channels at the sender. Every active[10] interface will get a packet from the queue. Packets to be retransmitted are also put back on the queue. On the receiving end, packets are put in order before passing them to the application.

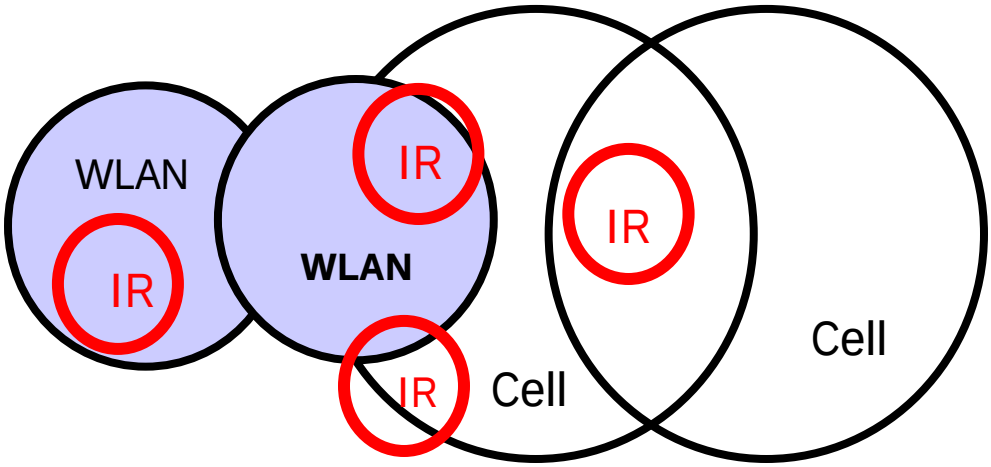
## *2.3 Link Layer Manager*

With the multitude of wireless technologies being deployed, little work has been done on how to create a generic interface to hide the details on how to configure each wireless interface to allow communication through it. The necessary steps go from finding the presence of that wireless infrastructure, to acquiring an IP address, passing through authentication and authorization. The role of the Link Layer Manager is to take care of

[10] An interface is active if it has link layer connectivity, an IP address associated with it, and its bandwidth/delay product allows it to be used.

the necessary configurations to allow a changing set of wireless interfaces to be used, possibly concurrently, by a mobile host.

In the wired world, Ethernet [DI080] is synonymous with local area networks. The factors that made Ethernet dominate were above all price and availability rather than a clear technical advantage over its competitors. In fact, as network utilization increased, Token Ring could have been a better choice, because Ethernet loses a lot of bandwidth to collisions when utilization reaches the 60% mark. The Ethernet dominance on the LANs spreads to other areas. Ethernet encapsulation is used in broadband MANs such as ADSL




(PPPoE [MA099]) and wireless.

**Figure 3: Example wireless coverage**

It is possible that in the future, Ethernet comes to dominate the Ether. However, nowadays there is no one-size-fits-all solution to wireless access. Radio technology has trade-offs in coverage and power, and this trade-offs may require specialized

technologies for the short, mid and long ranges. In addition, although infrared has never known widespread use, it has some nice characteristics like directionality, security[11] and price that may make it viable. Finally, the only technology that has planet-wide coverage is cellular. So in the near future, we envision that wireless connectivity will be a series of overlapping areas. In some areas, multiple technologies will be present. Figure 3 shows an example of overlapping coverage of many different technologies.

If a mobile host is navigating through the scenario depicted in Figure 3, it will find areas with different wireless coverage. In the proposed architecture, it has to acquire an IP address belonging to one or more networks in his vicinity to be able to communicate. The Link Layer Manager is the entity that is responsible for finding available communication resources and for acquiring an IP address.

Each link-layer has a discovery mechanism built-in into in. The work reduces to building a homogeneous API to hide the diversity of link layer discovery methods, to allow accessing all different link-layer discovery mechanisms using the same code no matter what type of link-layer will be used. In the future, we can envision the build up of a database where connectivity information will be kept. The mobile hosts will be able to query this database using the current available connectivity to find if other resources are available in the area. This may requires knowledge of spatial (physical) location, but the

[11] infrared does not go through walls, being harder to eavesdrop

mapping algorithms that allow a host know its location based on triangularization have been studied, for instance, in the Cricket Project [NI000].

It may be possible to reduce the amount of power used for link-layer discovery by using the last method, because the host will only power-up the interfaces that have base-stations on the vicinity of the host. Some wireless interfaces spend a significant amount of power just beaconing their presence, e.g., infrared interfaces. By keeping these interfaces powered down when there are no access points in the vicinity reduces the power requirements for the mobile. The requirements of building an infrastructure to maintain accessibility information and having the host know its physical location may not allow immediate deployment of that solution. The other way to conserve power is to change the search frequency of the interfaces. In the macro level, this can be done by powering down interfaces where there is no wake-up penalty in terms of power, and changing the duty-cycle of active search according to communication needs of the host. In the micro-level, indirect measurements of traffic (putting broadcast interfaces in promiscuous mode and listening for packets) and changing the periodicity of presence beaconing may be possible. Some link-layer protocols set the beaconing periodicity on their specification, so changing this periodicity may violate protocol specs.

Once the available link-layers have been identified and the user authenticated, an IP address has to be associated with the interface before it is used. For point-to-point links this means establishing an exclusive connection with the access point, which may not be desired. For some link-layers, the link layer manager can acquire addresses on demand.

This adds some latency to the communication, because the address negotiation will only happen after some packet has been sent through that link. A fall through solution also has to be present, if the link is no longer available (e.g. some other host is using the base-station).

Network configuration on IP networks is not an easy task. Until recently, a minimal configuration entailed getting a network number, a network mask, a default router and a DNS server, besides changing the DNS database to reflect the new host. To make configuration easier, and to allow for network number reuse for hosts that are not permanently connected to the network, a protocol for network configuration was created. The Dynamic Host Configuration Protocol (DHCP [DR097]) allows hosts to get a *lease* on a network address. This lease has to be renewed on a regular basis, or the mapping expires and the address can be reused.

DHCP has made configuration easy for dial-up hosts and for network configuration at large, because once a database is set up with the range of addresses that can be leased, adding a new host to a network is just a matter of connecting a cable. For mobile hosts, on the other hand, this model does not hold without changes. The problem is that, while for wired hosts the access to the network medium requires access to the premises (allowing some form of security), wireless access is not bound by walls. So for wireless hosts an additional step, which is link layer authentication, is normally added before DHCP can be used, making host configuration more complex. Additionally, there is no standard to control wireless link layer access. This means that we are thrown back to the

same stage we were before DHCP. For each wireless link layer that a mobile host uses, an ad-hoc, manual configuration has to be performed. The Link Layer Manager adds a general mechanism for link layer configuration, so that the operating system or an application can query what link layers are available, what are their characteristics and perform an automatic configuration on behalf of the user.

The Link Layer Manager is responsible for finding what communication resources are available in the current environment the mobile host is located, for acquiring an IP address for each available link layer, which may require authentication, and for notifying each instance of the protocols in use that a new communication layer is available. The LLM is also responsible for notifying the transport protocols that a link layer is no longer available, and that communication through it should stop. The basic services offered by the Link Layer Manager are summarized below:

1) **Link layer management**: a management entity can use direct information (by probing or listening to the link layer for the presence of access points) or indirect information (by using an existing connection to query the infrastructure for the existence of additional access points) to find new access points. This is called *link layer discovery*. Management also encompasses measuring signal strength and possibly location hints to rule that a link layer is no longer usable. This is called *link layer disconnection.*
2) **Network layer management**: before using a link layer, the mobile has to acquire an IP address for that interface. The most common protocol for acquiring a

network address in broadcast media is DHCP (Dynamic Host Configuration Protocol). For point-to-point links, such as infrared, acquiring a network address also entails creating a point-to-point link. In this case, the link will only be created on demand, as creating the link precludes other mobiles from using the same access point.

3) **Transport layer notification**: the transport layer has to be notified of new access points (in the form of a new IP address it can use) and of the loss of an active access point (an IP that can no longer be used). The transport protocols can also notify a management entity about the available bandwidth of each link. Because this bandwidth is closely tied with the available bandwidth of the last hop, by controlling the maximum bandwidth each protocol instance can use the management entity to enforce usage policies for cooperating protocols.

## *2.4 Location Service*

In the previous two Sections, we have shown how to take advantage of multiple link layers, and how to get access to these link layers. Both mechanisms run at the mobile, one being a multiplexing, mobility aware protocol, and another the Link Layer Manager. In this Section, we will present the third component of the architecture, the location service, which runs in the network infrastructure. Because our mobility architecture breaks an invariant in communication, which is the destination IP address, a corresponding host can no longer rely on the mobile's IP address to send packets to it. The location service is used to restore this invariant, by creating a way to find a mobile

host even when it is no longer at its home network. The challenge is to create a mechanism that is both reliable and scalable, to support the growing number of mobile hosts.

To understand the role of the location service in this mobility architecture, let's analyze how a connection from a corresponding (wired) host to a mobile host is established. Assume that a mobile host can be reached through two different types of addresses (similar to Mobile IP). The first type of address is the canonical host address given to the mobile host in its home network. While the mobile is in its home network, it can be reached (IP-wise) using this address. As the mobile moves away from its home network, it acquires a series of temporary IP addresses associated with the network or infrastructure the mobile is current using, by using the LLM. Communication to a mobile away from the home network has to be made ultimately using the temporary addresses. If a corresponding host is not aware of mobility, the only way for it to communicate with the mobile host is through its canonical address. A mobility-aware corresponding host, on the other hand, can use the location service to find the mobile host's new address, and use the new address for communication.

Because the proposed architecture assumes that mobility mechanisms will be implemented by the end-points, if a corresponding host is not mobility-aware it will not be running the mobility-aware transport protocols by default. But it is still possible to do bandwidth aggregation, albeit at some cost. A proxy located at the home network can do tunneling using one of the multiplexing transport protocols, as is done in Mobile IP.

Using a proxy is more wasteful of network resources, but it is still more advantageous to use a proxy because bandwidth aggregation can still be done, in contrast with a solution relying only on Mobile IP. However, a protocol suitable for encapsulation has not yet been developed. The architecture of mobility with a proxy for bandwidth aggregation is similar to the one presented in [SH004].

Generally, for direct communication, packets sent to the mobile host must be addressed to one of its temporary IP addresses. The set of temporary IP addresses associated with a mobile host will be constantly changing as the mobile hosts enters and leaves the coverage area of the technology associated with each address. When a corresponding host begins communication with the mobile host, it will have a name or the original IP address of the mobile host. This must be translated to one member of the set of temporary addresses the mobile host currently has. If communication begins while the mobile host is on its home network, the change in attachment points can be communicated in-band to the corresponding host, using any of the available connections already established. On the other hand, if the mobile host is away from the home network when a corresponding host[12] wants to initiate communication, another mechanism has to be used.

Two solutions can be used to address the problem of finding the mobile host. One is to use Mobile IP infrastructure. Mobile IP can be used to start communication, by giving one channel to the mobile that can be used to divulge the other local addresses. Using

[12] If the communication is terminated on a host in a fixed network, this problem goes away, because the mobile can send its address during startup

Mobile IP, no additional services have to be created. The infrastructure created for Mobile IP that is in place can be used, and using the multiplexing transport protocols can circumvent the limitations of Mobile IP. DNS then will work unchanged.

Another solution is to use a naming service. In [SN000], a dynamic DNS was used to find the mobile hosts, and in [WE099] the Session Initiation Protocol (SIP [HA099]) was used to create the bindings between a name and a mobile host. If a DNS solution is used, a dynamic DNS server must be constructed to keep track of the set of addresses that the mobile host acquires as it moves. As a query is made, the DNS server answers with the set of temporary addresses. The corresponding host can then communicate directly with the mobile. After initial contact is established, the mobile reports any changes in the set of addresses using the mechanisms built into the transport protocol.

There is still a hybrid solution using both dynamic DNS and a proxy on the home network. The DNS query in this case will return the canonical address of the mobile host, and a home agent can either serve as a proxy and forward traffic to the mobile, or redirect the traffic to the mobile by sending a redirect message to the corresponding host with the set of temporary addresses the mobile is currently using. The idea of redirecting the traffic to a new address was explored in [SN001] where TCP's bindings were changed to from the 4-tuple (sender-IP, sender-port, destination-IP, destination-port) to a triple (sender-IP, sender-port, connection identifier). The connection identifier can keep a connection working even if the address of the mobile changes. In [XI002] , this work was developed further by allowing both sides of the connection to be mobile.

It is clear that using a DNS solution makes the prototyping very easy, because most of the DNS resolver code can be used without any changes. However, because the dynamic server must be in constant contact with the mobile, and track each addition and deletion of IP addresses, this solution may not scale. Of course, the major scaling problem is not updating the server. The DNS infrastructure works well because the databases are mostly static, and aggressive caching can be done to minimize DNS traffic. Because addresses associated with mobile hosts are temporary, no caching can be done. There are mechanisms in DNS to prevent caching. However, turning off caching may cause a large performance hit, since every query must be sent to the DNS server of the home domain.

The proxy solution puts the burden of tracking the mobile on the home agent. If the home agent is to serve as a proxy, it must track the location of the mobile, so in the end this may be the simpler approach. Because Mobile IP has had an extensive deployment, being built-in in all Cisco Routers since IOS 12.1, the proxy approach using Mobile IP can be implemented immediately. A dedicated aggregation proxy requires more work, but can be used in the absence of the Mobile IP infrastructure.

## *2.5 Conclusions and Future Research*

The main contribution of this chapter is the architecture for transport layer based mobility. This architecture allows the reuse of the IP infrastructure that is in place without modifications. It is also more efficient than Mobile IP, because it does not need tunneling and triangular routing. The architecture also allows for two characteristics that

are already present in the current mobility infrastructure: that link layers for mobile hosts are wireless, and that in many areas there are multiple overlapping wireless technologies available. These were taken in consideration in the design of the transport protocols that are the centerpiece of the architecture. The protocols are tuned for wireless links, which means that they have a congestion control mechanism that is well suited for wireless communication (specially in regard to the higher bit error rate found in wireless). They can also distribute the data being sent into all available links, by using inverse multiplexing. The application sees one channel, but the protocol uses many sub-channels, each one mapped to an interface, and does load balancing and resequencing as necessary.

There are many possible extensions to the ideas proposed in this Chapter. The Link Layer Manager, which was partially implemented in [TO002] can be augmented by a “World Map”. The World Map is envisioned as a distributed connectivity database. It can be cooperative, in the sense that each mobile host can contribute to the database by updating it with the connectivity information seen by the mobile. The idea is to allow the mobile host to save energy by avoiding to probe for non-existing infrastructure, or to make handoffs faster by knowing what infrastructure is present. When a mobile host is moving to a location, it can update is “available infrastructure” database with the current “World Map” information. The mobile host then can know what infrastructure is most likely to exist in a certain place, and can stop or diminish its search for alternative wireless links.

Another area of future research is the usage patterns of wireless hosts. There is already research in this area [KO002], but it has been limited to campus environments, where

movement tends to follow certain patterns. It is unknown if movements of the general population can be inferred from student movement in a campus. Cellular companies can have access to this information, by tracking the user identification code of each cell phone, but privacy issues tend to make this information unavailable for research.

Finally, location service has had an enormous increase in interest due to Voice over IP (VoIP [RO000], [DA000]). The Session Initiation Protocol (SIP [HA099]) can be an interesting alternative for the tracking of a completely mobile host (one that has no home address). How scalable SIP to support constant updates remains to be seen.

# Chapter 3 Congestion Control

## 3.1 Introduction

This Chapter contains the description and evaluation of the congestion control algorithm called Homeostatic Congestion Control (HCC), developed for the suite of rate-based, mobility-aware transport protocols. A new congestion control algorithm was developed to help achieve bandwidth aggregation. HCC measures available bandwidth, which is used in turn as input to the load-balancing algorithm that divides a flow in the (possibly) multiple available (wireless network) interfaces. Beyond keeping the transmission rate of a protocol below the network congestion point, the constraints imposed on a congestion control algorithm are convergence to available bandwidth, which measures how well a protocol uses available network resources; its stability by itself and in the presence of competing traffic, which assures that the throughput of a protocol does not oscillate in response to external stimuli; and fairness, which is how well bandwidth is divided between different flows. This Chapter describes the design decisions to achieve such goals and the evaluation of how well HCC is able to conform to these requirements.

## 3.2 Background

Congestion control is important for the well being of the network infrastructure. The earliest version of the Transmission Control Protocol [PO081] did not have congestion control [NA084]. Packets were sent in bursts of one flow control window at a time.

Although this worked when the network was lightly loaded, the result was poor throughput as network usage increased. Lost packets caused retransmissions that in turn tended to further congest the link, generating more lost packets in a vicious circle, until network capacity was completely taken by retransmissions, and very little data got through. Congestion control was added to TCP by Jacobson [JA088], and the innumerous flavors of TCP differ in general in the algorithms used for congestion control, with some exceptions[13].

Today, besides maintaining good network health, new congestion control algorithms must also be good neighbors. It is important that a new protocol does not take more bandwidth than TCP, which became the metric because it is the protocol most widely used for transport in the Internet [CL000]. This concept of maintaining the same throughput that TCP would under the same network conditions is called TCP-friendliness. It has been formalized by Sally Floyd in [FL000].

The main contribution of this chapter is the definition of the Homeostatic Congestion Control, which is suitable for rate-based protocols. HCC uses packet pairs and inter-arrival times for bandwidth measurement that translates into the rate packets will be sent. In contrast, another rate-based transport protocol, RAP (Rate Adaptation Protocol [RE099]) uses additive increase, multiplicative decrease (AIMD) for congestion control.

[13] It is questionable whether the selective acknowledgements, for instance, can be considered a change in congestion control, in the sense that it does not change the timing or the amount of data sent. SACK does allow for better selection of retransmissions, which in turn makes TCP more efficient.

AIMD has been shown to converge ([CA004] [LA002] [LO003] [ZH004]) and it is a good alternative to HCC, although the secondary constraint of dividing a flow into multiple paths led to a preference on bandwidth measurements in our protocol architecture. Other protocols deal with the problem of TCP-friendliness directly, e.g. the TCP Friendly Rate Control Protocol (TFRCP [HA003]) uses the actual analytic formula for TCP throughput to limit the sending rate of the protocol, which can guarantee that the protocol will not grab more bandwidth than TCP under similar delay conditions.

This Chapter begins describing the general problem of congestion control in Section 2, presents the rate-based mechanisms in Section 3, how congestion can be inferred by the algorithm in Section 4, how to adapt to varying network conditions in Section 5, the central idea of Homeostatic Control in section 6, the Homeostatic Congestion Control Algorithm in Section 7, the validation on Section 8 and finalizes with the conclusions on Section 9.

## *3.3 Bandwidth Estimation, Congestion Detection and Control*

The central question of a congestion control algorithm is how to detect if congestion is happening. Due to the reliability of most physical links in the Internet, lost packets can be used as a congestion indication. The receiver can infer the sender is overwhelming the buffer space of a router in the path, that is, it is pumping more packets into the network than the bottleneck link can forward when a packet is lost. That loss can then be reported back to the sender, and adequate measures taken, namely, lowering the sending rate to

stop the congestion. This heuristic is not devoid of problems. The most obvious is that not all packet drops are caused by congestion, especially when wireless links without link layer reliability are being used. Another problem is the time lag between congestion and prevention. The receiver must notice the loss, which normally entails receiving one or more packets after the loss event, or the expiration of a timer. The loss must then be reported back to the sender, which requires half an RTT. An approach to minimize the lag is to use Explicit Congestion Notification (ECN [RA001] [MO003] [MO103]), in which congested routers warn hosts[14] that packets are being dropped. The reasoning behind deploying ECN in routers is that the magnitude of the time lag between when a loss occurs and when the sender notices and reacts to it can be large. Meanwhile, the sender continues to pump more data in the network than the network is able to transfer, which aggravates congestion.

Ideally, a sender would know how much data the network is able to transfer, that is, the available bandwidth in a network path. Without reservations, unfortunately, it is almost impossible to know beforehand how much bandwidth is available, because this amount varies in an unpredictable fashion. TCP's congestion control is a form of bandwidth measurement. Although the algorithm converges, it does not measure the available bandwidth, but instead the sum of network bandwidth and routers' queues. Furthermore, it requires straining the network (by causing losses) to find network limits. The elasticity

[14] Routers may notify senders by using ICMP packets, or receivers by tagging packets when congestion is present.

of the routers, designed to accept bursty traffic, is in part responsible for the inaccuracy of the measurement. To cause a loss by overflowing the router queues, not only the incoming packet rate has to be larger than the outgoing packet rate, but also this situation has to last long enough to fill the router's outgoing queues. The result is that TCP overestimates the path bandwidth, and keeps causing losses even in stable conditions. In response, the challenge is to measure the instantaneous available bandwidth without causing congestion on the network.

To find out what congestion indicators can be used in place or to supplement packet drops, it is necessary to analyze packet dynamics [PA097]. The first step is to find if packet loss is a good congestion indicator, by classifying losses as caused by problems with transmission or by congestion. Discriminating whether losses are caused by congestion or by the network is complex and end-to-end statistics do not have good correlation with loss type [BI099]. Therefore, the converse problem, which is detecting congestion using end-to-end statistics, is also hard. If end-to-end information does not have statistical correlation with loss type, that is, even after detecting a loss it is hard to tell if this loss was caused by congestion or by transmission errors, then it is hard to predict if congestion will happen by looking at end-point information such as interarrival time. To improve the correlation between interarrival time and network conditions, a rate-based transmission mechanism can be used as part of the bandwidth measurement mechanism. Transmitting packets at regular intervals prevents the burstiness inherent to an ack-clocked design, limiting the amount of variance in the transmission of packets and

creating a flow with characteristics better suited to analysis.

## 3.4 Rate Based Mechanisms

To achieve the goal of bandwidth aggregation by load-balancing a flow through multiple interfaces, we decided to use a rate-based mechanism in our protocols. The rate-based flow generates a better correlation of interarrival times and network conditions, and allows for better bandwidth estimation. In contrast to ack-clocked transmission mechanisms, where packet transmission time is dictated by the arrival of acknowledgements from the receiver, a rate-based transmission mechanism transmits packets at regular intervals, specified by the current packet rate. A rate-based transmission mechanism therefore produces traffic similar to a sequence of CBR streams. The difference between each CBR stream and its predecessor is in their transmission rate. Depending on the algorithm used for adaptation to the network conditions, different thresholds on observed conditions will be used to signal the end of a stream and the beginning of another, which uses a new rate adapted to the current conditions.

The dynamics of a CBR stream can be analyzed and generalized to a time-varying stream consisting of a series of consecutive CBR streams with different rates. If there is no intervening traffic, the time dependencies between any two consecutive packets of the CBR stream should stay constant no matter what pair is chosen, excluding boundary effects when the rate changes. If the CBR stream rate is below the service rate of the bottleneck link (in this case defined as the link with lowest capacity), there is no queueing

at any of the intervening routers. Moreover, each packet after the first encounters the same conditions in each and every router in the path, so the total transit time between source and sink is the same for every packet. On the other hand, when the rate is above that of the bottleneck link, a queue starts forming at this link, and the rate seen at the receiver becomes the service rate at the bottleneck link.

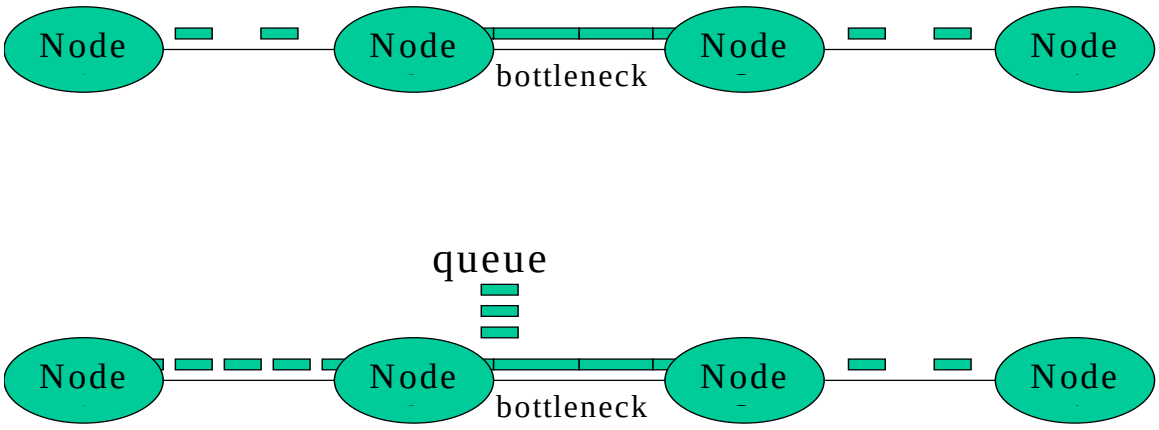



**Figure 4: If the sending rate is below the service rate at the bottleneck link, the timings of the packets are preserved, otherwise a queue forms and packets are delayed**

Therefore, in the case of no intervening traffic, feeding back the rate observed at the receiver to the sender, and adjusting the rate at the sender if it is above the rate observed at the receiver can control the rate. Of course this adjustment mechanism only works if the rate is overestimated: if the rate used is below that of the bottleneck link, no change will be perceived at the receiver. One way to find the real bottleneck capacity is to send packets at the sender's maximum rate (back-to-back) and measure their interarrival time.

The interarrival time is the dispersion caused by the bottleneck link. If packets are sent at the rate that produces this dispersion no further delays will be caused at the bottleneck link, because the spacing of the packets corresponds to the service rate at the bottleneck link. The bottleneck is normally encountered on the first or last link, due to last mile problems, but the preceding argument works with any bottleneck location.

The above idea is the packet pair method [KE092], if fair queueing is used at routers to isolate each stream from variances of other streams. Using packet pairs and fair queueing, packet dispersion can be used to measure the bottleneck link, or the lowest link capacity in a network path. With no competing traffic, the queueing policy at the routers is not important, since it will have no effect on the dynamics of the pair.

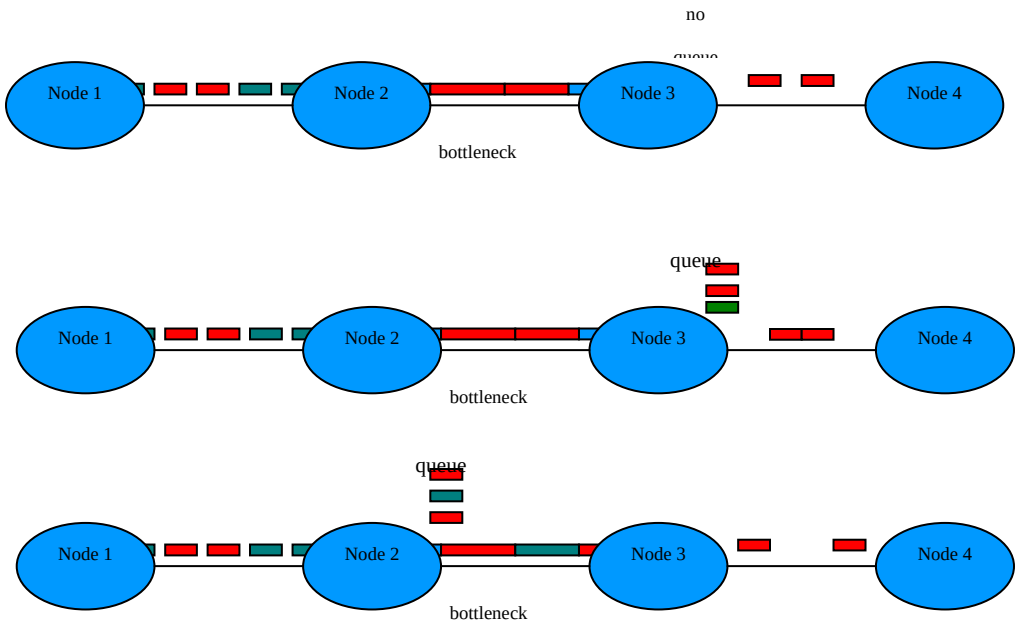



**Figure 5: The arrival time of two consecutive packets can change depending on the network conditions: if there is no queueing after the bottleneck link, this will correlate with the bandwidth of the bottleneck link**

When there is intervening traffic, the queueing policy in the routers is very important. If fair queueing is used, then each flow will get its fair share of bandwidth. Fair queueing effectively isolates each flow from the effects of burstiness in other traffic. The packet pair will then measure the share of bandwidth allocated to the flow. On the other hand, if a FIFO policy is used (as is common in many Internet routers), a more complex behavior ensues. Two effects come into play, as can be seen in Figure 5: time compression, when there is a queue in a router and the first packet gets delayed; and time expansion, when one or more packets get between the first and second packets in the packet pair. Both effects can affect the same packet pair in different routers along a path, which generates a large dispersal in the interarrival values measured at the receiver.

## *3.5 Congestion Indicators*

When using packet pair to measure the bottleneck bandwidth, time compression and time expansion must be filtered out. Unfortunately, as shown by [DO001] the distribution of the interarrival times is multi-modal, and the main mode does not necessarily correspond to the bottleneck bandwidth. This lack of correspondence invalidates statistical filtering of the dispersal values to find the bottleneck capacity [BI098]. Moreover, the current tools to measure bandwidth require a long time to converge (in terms of round trip times) and therefore can only measure capacity, and not the current state of the network, which is the relevant information in this instance.

To use bandwidth measurements as a congestion control tool, it is necessary to know

instantaneous network conditions. This is, of course, impossible. The best alternative is to use information gathered at the receiver and sent back to the sender. This introduces a delay between the measurement and the use of the information, which makes the information only a hint of the current network conditions. Therefore, the mechanism must be inherently dynamic. The predictions will be inaccurate most of the time, because the network conditions will normally change from the time the measurements were made and the present. However, if they can be made good enough, and a mechanism to correct the effects of the low accuracy is present, then the tool will be effective. It must be pointed out that all transport protocols follow this principle. The network conditions will always be unknown at the sender[15], unless information can be gathered by the routers and fed back to the sender. Even in this case, the farther away the router is, the more outdated the information becomes, because it still must be fed back to the sender. Therefore, protocols need a heuristic to deal with the unknown varying network conditions, and a way to correct the failed predictions.

## *3.6 Congestion Control Mechanisms*

The basic mechanism chosen for bandwidth measurements in this case is packet dispersion, the time difference between arrivals of two packets. A single instance of packet pair does not give reliable information. It can underestimate or overestimate the

[15] The receiver, however, can gather information on network conditions recording metadata on the packets, such as loss, loss rate and arrival time.

fair share of bandwidth. To accurately track available bandwidth, two variants of the packet dispersion algorithms are used: the first is the above cited packet pair method, where two packets are sent back-to-back, and the second is the regular measurements of jitter, the difference between the interarrival times of packets at the receiver and the rate with which they were sent. This is more meaningful due to the rate-based sending mechanism, where packets are sent at a known regular rate. In both cases, the packet dispersion is measured. The first technique has been used extensively to measure the minimum capacity link in a path (and not fair share), but requires many repeated experiments and statistical analysis ([LA001] [DO099] [LA000] [CA096] [BO093] [LI004] [HA001] ). The second technique has also been used for capacity, with similar requirements [AH099].

The objective of the dual mechanisms is to achieve homeostasis, where two opposite forces balance out in a dynamic equilibrium. The operational point which must be achieved, called “fair share”, is such that connections with equal characteristics will get the same share in the critical routers, and connections with different characteristics will not be starved out. The critical routers are those that have low capacity due to actual link bandwidth or heavy traffic.

The description of the two mechanisms that follows assumes that FIFO queueing is being used at the routers. The first mechanism, packet pair, produces a bandwidth measurement at the receiver. This measurement is passed through a tunable low pass filter to smooth out transitions and prevent oscillations. A randomizer is used to prevent synchronization,

which can be a problem to rate based protocols [AG000]. The resulting value is sent back to the sender and used as the new rate. If the new rate value is below the “fair share”, the interarrival times will tend to be dispersed, and positive and negative values of jitter will be received. This happens because the critical router is not working at maximum capacity. Sometimes the packet pair will be queued together, and the resulting value of jitter will be negative as they arrive at a lower interval than the sending rate; sometimes the packets will be separated, and if this time separation is greater that the rate the jitter will be positive. However, if the new rate is above the “fair share”, there will be a bias towards positive jitter values, because after the critical router the packets will tend to be spaced farther apart than they were sent. When a threshold of consecutive positive jitter values is seen, it serves as an indication that the current rate has overshot the accepted “fair share” rate. The receiver requests a rate decrease, using the average value of the jitter as a measurement of the amount of overshoot in the rate.

It is important to notice that the “fair share” concept does not guarantee that protocols with different congestion control mechanisms will get equal throughput. Even among TCP connections, if connections have different RTTs, they will not get the same bandwidth on the links they share [KL004]. Connections with lower RTTs will tend to get a greater share of bandwidth, even everything else being the same. TCP friendliness is normally defined using the simplified equation for TCP throughput [MA097]], where throughput is equal to a constant C (1.22 or 1.31 depending whether delayed acks are used) divided by the RTT times the square root of expected loss events (loss).

(1) Throughput = C/(RTT * √ loss)

This is valid for some very constrained conditions, namely that the error rate (meaning the drop rate) is below 5%, so that no timeouts occur. In the general case, the formula is much more complex, and depends also on the receiver's declared window size, and the base timeout value [PA099].

It is possible to create protocols that obey expression 1 or the more complex equation. This can be done by using the calculated throughput as a cut-off value for maximum bandwidth, and not exceeding this value in any time interval. This was done in [DO000] using the simpler expression for throughput and in [PA099] using the full dependencies (see also [HA003] [FL000]). Unfortunately, the network dynamics are complex enough that following the cutoff expressions will not guarantee that the protocols will adapt in a TCP-like fashion. In fact, it is clear that to adapt in a TCP-like fashion the protocol must use the same transmission strategy that TCP uses, which means to be TCP. In fact, even different flavors of TCP will not be TCP-friendly among them. According to the network conditions, different flavors get an advantage and grab more bandwidth than their brethren.

## *3.7 Adapting to dynamic conditions*

As will be shown below, three parameters can be altered to change the behavior of protocols using the proposed mechanism. The first is gain in the low pass filter used on the rate calculation after a packet pair is received. The gain regulates how much history is

kept when a new rate is calculated. The gain of the filter can be adjusted so the protocol adapts slower or faster to the current network conditions. While it would be nice to adapt as fast as possible, every rate adjustment has influence on all traffic on that path, including the adapting flow. If the value overshoots the desired operational point, it is possible that the next adjustment will try to correct this, possibly creating oscillations. The second parameter is in the jitter monitoring mechanism. If the rate overshoots the operational point, this mechanism notices positive jitters, and adapts the rate. Ideally, the rate should be dropped to the operational point. A hint of how much error the current rate contains is given by the jitter values. A value proportional to the average jitter is used to correct the rate. The threshold (how many consecutive positive jitters) can be adjusted, and how much of the jitter is used to correct the rate can be adjusted so that the protocol backs-off faster or slower, and more or less in case of errors in the "fair share" rate calculation. The third parameter is the amount of back off due to congestion. This defines how much the rate has to be decreased in case of packet losses. TCP uses a multiplicative decrease, halving its window in case of losses. A similar approach is used, and the rate is halved in case of losses. Although halving the window and rate are not exactly equal, the effect is the same, namely, to back off enough so the queues that have been building up in the network have a chance to clear. If every protocol used the same technique, it would be possible to back off less, but in the current conditions, it is safe to err in the side of caution and back off more aggressively.

Other protocols have used similar approaches. Wireless TCP (WTCP) [SI099] is a split connection protocol that tries to find an operational point by keeping jitter histories.

Although this approach is promising, WTCP was designed for low bandwidth CDPD networks, where traffic is not high and the hop count between the proxy and mobile host is low. In more general networks, it is very hard to find the operational point. It is not generally possible to know network conditions at the start of the connection, and the network conditions are needed to define the operational point. The Rate Adjustment Protocol (RAP) [RE099] shares with WTCP the rate based approach, but uses an AIMD (Additive Increase, Multiplicative Decrease) mechanism for congestion control.

## *3.8 Homeostatic Control*

In this section the details of the congestion control heuristics are given. The name homeostatic control comes from the two mechanisms used for achieving balance. The heuristic tries to keep balance by overestimating the available bandwidth (due to the bias of the packet pair to measure link capacity and not available bandwidth), and correcting the estimate by slowing down the rate using jitter averages.

### 3.8.1 Rate based sending mechanism and jitter monitoring

Using a rate-based sending mechanism means that packets are sent at a certain *rate*, or conversely, that packets are sent with a certain *period*. Due to flow conservation, the number of packets that enter the network must be equal to the number of packets that leave the network. Packets may leave the network either by arriving at the destination or by being dropped at one of the routers in the path. In a best effort network, packets are only dropped because of errors in the header field (caused for example by bit errors in

transmission) or because there are no buffers available at the router output queue. In the absence of transmission errors, the main cause for lost packets is congestion.

The existence of buffer queues in the routers means that it is possible to violate the maximum service rate for a limited time without adverse effects. In fact, TCP does that all the time, by sending bursts of packets that are queued at the routers for transmission. On the other hand, if the bursts are large enough, it is possible to lose packets even if the average packet rate is below the service rate of all routers in the path, because the burst may temporarily overflow the available buffering capacity. If packets are sent evenly spaced, this possibility can be ruled out. This is the idea behind TCP pacing, which unfortunately may have worse throughput due to flow synchronization [KE000].

If the transit times of all packets were the same, they should keep the same timing relation with which they were sent. Unfortunately, due to the presence of cross traffic, even packets sent at a rate below the maximum service rate of a path will have different transit times (packets sent above the maximum service rate will certainly be delayed, and if the rate is kept for long enough, some of them will be lost). If a packet is delayed in relation to the previous packet, the receiver notices a positive jitter. If jitter is calculated taking in account the difference in arrival time between two consecutive packets, and their original time difference when transmitted, negative jitters are also possible. If the first packet is delayed, and the second does not suffer any delays, they arrive closer together (time wise) than they were sent. Calling jitter the difference between their interarrival time and the period with which they were sent, the result is negative jitter. In

Figure 6 we see how positive and negative jitters are generated.

Positive and negative jitters should alternate. When a series of positive jitters is noticed, that means that the network service rate is being violated. The average jitter is a good approximation of the difference between the feasible sending period and the current value of the period. When using homeostatic control, two consecutive positive jitters are taken as an indication of rate violation, and the period is corrected to a value that should result in a sending rate that is below the network service rate, given the conditions observed by those two packets.

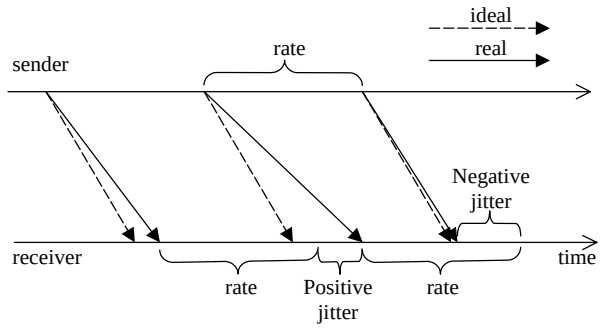



**Figure 6: positive and negative jitter**

Jitter monitoring prevents the period from straying from the network operational point. Unfortunately, this is not enough to guarantee that the network will operate with no losses. It is possible that small differences will build up eventually, even when positive and negative jitters are perfectly intermingled. The correction mechanism should decrease the rate to a point below the operational point, allowing any queues that have formed to shrink. This is also a good place to add some randomness. If different rate-base flows become synchronized, they will increase and decrease their rates in tandem. While

this may increase network utilization for a short time, if all flows violate the network service rate at the same time, they may all experience simultaneous losses. If those losses are coupled with a backoff mechanism, then all flows will backoff at the same time, leading to a large waste in bandwidth. By varying the amount of rate decrease using a random function, different flows avoid becoming synchronized, at the cost of never achieving full link bandwidth.

It must be pointed out that the disappointing results of TCP pacing are directly related to synchronized losses. By adding randomness in the rate calculation, synchronization is avoided. On the other hand, in order to prevent using a higher rate than what the network can support, the resulting sum of calculated rate plus the random factor will always be smaller or equal the maximum permissible rate, never higher. Thus the theoretical impossibility of achieving full link bandwidth.

### 3.8.2 Bandwidth estimation using packet pair

Previously in the chapter, the flow of a rate-based protocol was compared to a sequence of CBR streams. The advantage of a rate-based protocol over an ack-clocked protocol for bandwidth measurement is the better correlation between interarrival times and network conditions. The more regular the flow is, the better correlation is achieved. A CBR flow would be the perfect flow for measuring network conditions. Unfortunately, a CBR flow lacks a mechanism for adjusting the rate upwards. While it is possible to reduce the rate by watching an increase in interarrival times, it is impossible to guess how much

bandwidth is available without actively probing the network.

Therefore, the rate controller has to probe the network from time to time to find if more bandwidth is available. However, probing destroys the regularity of the flow, and poisons the statistical data being gathered. The challenge is to probe often enough so the flow can react to network conditions in a timely manner, and not so often that no useful information can be extracted from the flow.

Empirically, we found that probing more often than once every three packets had adverse effects over the correlation between interarrival time and network loading. Therefore, to make the mechanism as responsive as possible, it was decided to create trains of five packets. Every four packets, the fifth is sent back to back with the fourth packet in the train. These last two packets are called “probe packets”, as they are meant to allow the measurement of the available bandwidth. The interarrival time of the probe packets is measured, and this is an indication of the minimum time separation that the network can achieve.

Undeniably, there is a tendency that the packet-pair will overestimate the available bandwidth, because unless the competing flows are also rate-based the packet pair may measure link capacity, and not available bandwidth. To prevent a large change, the new value measured for the period is mixed with the current value. How much history is kept, and how much of the new value is used can be tuned. This controls how fast the algorithm will respond to change. The less history is kept, the faster the algorithm will converge to a new value. On the other hand, fast changes can lead to oscillations, and

ultimately to congestion.

The same arguments against synchronization apply to bandwidth estimation, and the same solution can be applied. A random percentage of the measured interarrival time is added to the period calculation. That means that the algorithm will rarely achieve full path bandwidth, being cutoff from the maximum exactly half of the random function. The magnitude of the random function was chosen empirically to balance performance with friendliness to other flows.

## *3.9 Algorithm*

There are 3 phases in the algorithm:

1. Exponential increase
2. Congestion avoidance
3. Congestion Control

In the exponential increase phase the idea is to converge quickly to the available bandwidth, so packet pairs are sent once every 5 packets. The period in which packets are sent is adjusted for each measurement according to equation (1) below. The error, or the difference between the optimal period and the current period is given by equation (2) if the optimal period is constant, that is, if the available bandwidth is not changing. Therefore, the error decreases exponentially for a static scenario.

$$P_{n+1} = (1 - \alpha) * P_n + \alpha * P_{Measured} \quad (1)$$

$$\alpha \in [0,1]$$

$$Error = ((1 - \alpha)^n) * (P_0 - P_{Optimal}) \quad (2)$$

The congestion avoidance phase is signaled by a sequence of positive jitters, showing that the network is getting loaded, and queues are increasing. In that phase, the period is corrected by the error between the current period and the optimal period. The magnitude of the error is given by the sum of the jitters, because the jitter is caused by the difference between what was measured and what packets experienced while in the network. So the new period is calculated according to equation (3)

n - number of measurements

$$P_{n+1} = P_n + (jitter_1 + \ldots + jitter_n)/n \quad (3)$$

Where n is 2 or 3

In the third phase, congestion control, the protocol has noticed that the network is congested, by experiencing congestion related losses. The protocol then starts the multiplicative decrease, doubling the period every RTT following the expression below:

$$If\ (current\_time > time\_last\_loss + RTT + 2 * P_n)$$

$$P_{n+1} = P_n * 2$$

Let Pc be the current period, $P_{Measured}$ the measured period and $P_{Optimal}$ the true optimal period. We have 3 cases:

1. Too small : $P_{Measured} < (P_{Optimal} - (1-\alpha) * Pc) / \alpha$
2. Sweet spot: $P_{Optimal} > P_{Measured} > (P_{Optimal} - (1-\alpha) * Pc) / \alpha$
3. Too large: $P_{Measured} > P_{Optimal}$

If the period is too small (case 1), the rate will be large, and may cause congestion. If the period is too large (case3), the protocol will not achieve maximum throughput. If the period falls in the optimal area (case 2), the new period will be close to the one that will result in using all the available bandwidth of the link. The homeostatic nature of the congestion control acts to correct the errors in measurement. Because there is a bias to overestimate available bandwidth, normally the measurements will fall in case 1 or 2. Case 2 is optimal. Case 1 will generate positive jitters, and jitter correction will come into play, slowing down the rate. Case 3 is more rare, and only temporarily slows down the rate, until the next measurement.

## *3.10 Experimental section*

In this section, the results of simulations done using the ns-2 module that implements the congestion control mechanisms proposed for the new mobility-aware transport protocols are presented. While actual implementations are invaluable for adding real world constraints, many uncontrollable variables are introduced that may influence the outcome. Examples are the asymmetries in the base station backoff delay and station

backoff delay in the wireless Ethernet Rangelan, or the one slot buffer in the NetBeamIR infrared base station. Moreover, it is impossible to vary link capacity and delays incrementally without introducing artificial mechanisms (such as a delay adding proxy), which diminishes the reality of the experiment. Therefore, XMTP for ns-2 was developed, to isolate the testing of protocol mechanisms from environmental constraints. With the mechanisms tested through simulation, Chapters 5 and 6 present the description and testing of protocol implementations in real world environments.

The evaluation is designed to test six aspects of the homeostatic congestion control mechanism:

1- **Convergence to link bandwidth**: test to see if a XMTP flow use the full bandwidth of a single link. A single XMTP flow is run, and it should reach within 10% of link bandwidth in a short time

2- **Convergence to available bandwidth**: test XMTP ability to respond to changes in bandwidth. A single XMTP flow is run, and its response to a square wave is tested. The square wave is a CBR flow that lowers the available bandwidth and then ends, returning conditions to where they were in the beginning. XMTP should to lower its sending rate and resume its previous sending rate when the CBR flow ends

3- **Fairness**: test XMTP ability to share bandwidth with itself and other flows. Two experiments are run. First a small scale with five flows, so individual characteristics of each flow can be analyzed, and then a large

scale, with one hundred flows. The number of packets received is recorded, and Jain's Fairness Index [JA084] is calculated. XMTP should to be able to share bandwidth well with itself and TCP, and the fairness index should be close to 1.

4- **Stability**: test XMTP response when multiple flows are present. This is the same experiment as the one before, but now the objective is to measure the smallest and largest number of packets observed in each flow, and the standard deviation from the mean. The smaller the deviation the better the result.

5- **TCP Friendliness**: test the response of TCP in presence of XMTP flows. The experiment is the same done above, but now the objective is to observe if TCP is unduly affected by XMTP, and the percentage of the bandwidth TCP and XMTP are able to use.

6- **Performance**: test the ability of a group of XMTP flows to use the available bandwidth. It is the same experiment as above, but the objective is to record what percentage of bandwidth XMTP is able to acquire. XMTP should be within 10% of the bottleneck link capacity.

### 3.10.1 Convergence to Bottleneck Bandwidth

This experiment consists of a single flow with no cross traffic going through a bottleneck link. The delay of the bottleneck link is changed with each run. The results shown are for a single run for each delay. Although XMTP is not completely deterministic because

operations such as calculating the period have a random component to prevent synchronization, the runs are reproducible exactly if the same random seed is used. The number of packets that arrive at destination in each second is recorded. This number is then compared to baseline, which is a TCP flow under the same conditions.

To establish a baseline, the experiments are run using TCP New Reno for different link delays, shown on Figure 7. TCP New Reno has problems with high delays. That happens because, due to the burstiness of TCP traffic, the queue at the router fills up and Reno experiences losses even before it can fill up the bottleneck link. Therefore, for a certain bandwidth/delay product, if the queues at the routers are not large enough, TCP New Reno (and other flavors of TCP) cannot achieve the path throughput. This is coming into much evidence today in high bandwidth links, where the bandwidth delay product is high even for low delays. At higher RTTs TCP cannot achieve the maximum throughput of 124 packets per second, even though the only variable changed in each run was the link delay on the bottleneck link. Although is practice to scale the queues to the bandwidth delay product of the path (which would allow TCP to achieve link bandwidth for all delays) the queue size was not scaled to show one of the characteristics of XMTP, which is the low loading of the routers in the path due to the rate-based sending mechanism.

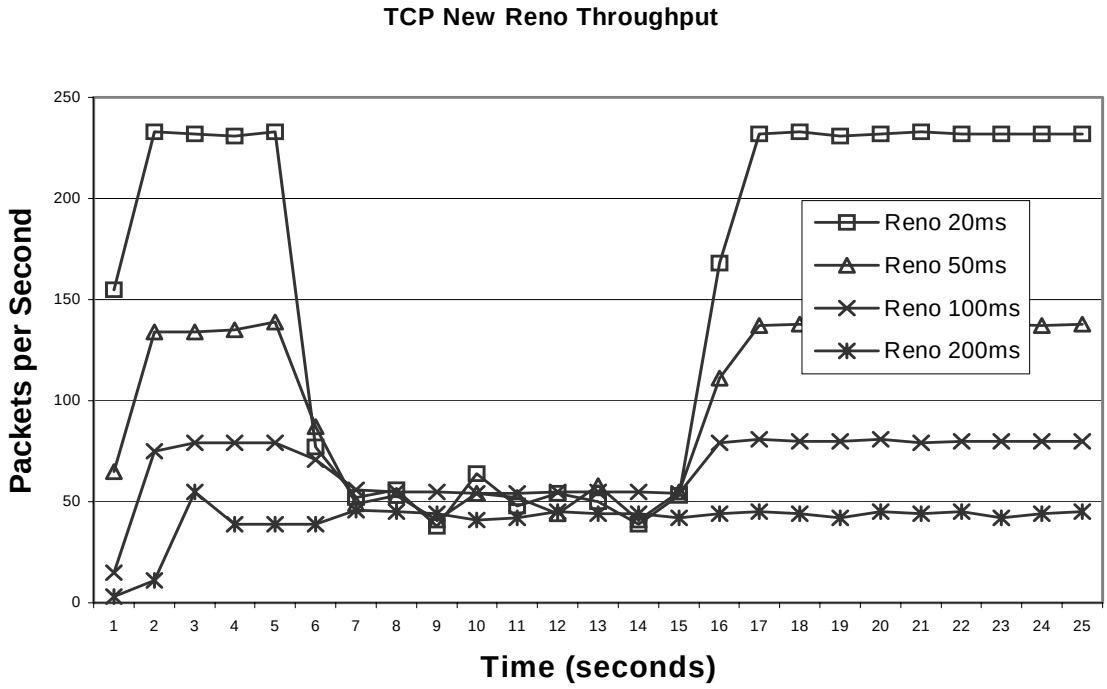



**Figure 7: TCP New Reno Throughput for Different Link Delays**

During the handshake phase of XMTP, an estimate of the bottleneck bandwidth is acquired by using the packet pair method. A single instance of the packet pair is not very accurate, and may overestimate the available bandwidth. To be conservative, the value obtained is doubled, so the initial value of the rate should be below that of the bottleneck link. The objective is to get an estimate that will be refined and changed during the connection lifetime, using the data gathered and taking in consideration the varying network conditions. In this experiment, because the environment is controlled, there is no cross traffic, so the value obtained should accurately measures the bottleneck bandwidth. Because the value is doubled, the rate will be half of the maximum rate available. Each subsequent measurement should increase the rate, until it converges to the available

bandwidth. The rate of increase is directly connected to the amount of history kept in each iteration of the probing packet pair.

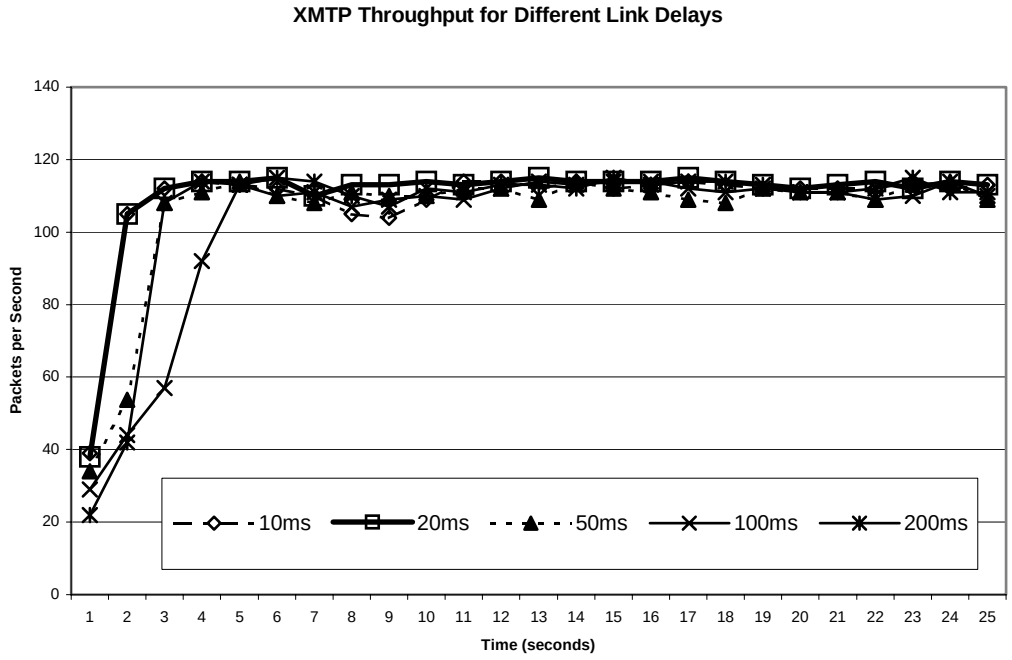



**Figure 8: XMTP Throughput for Different Link Delays**

In Figure 8. XMTP's throughput is shown for different link delays. Because XMTP is rate-based, it does not suffer from the problem of overflowing queues that prevent TCP from achieving the maximum throughput possible. Although XMTP does not achieve the maximum available bandwidth of 124 packets per second, the only influence on the link delay is the ramp-up time. Flows with different bottleneck delays still achieve the same throughput. The effect of randomness can be seen as the throughput fluctuation during "steady-state," and also on the 200ms run, where ramp-up was faster than the 100ms run. This random variance was introduced to avoid synchronization, and results in the

fluctuations below link bandwidth seen on the graph.

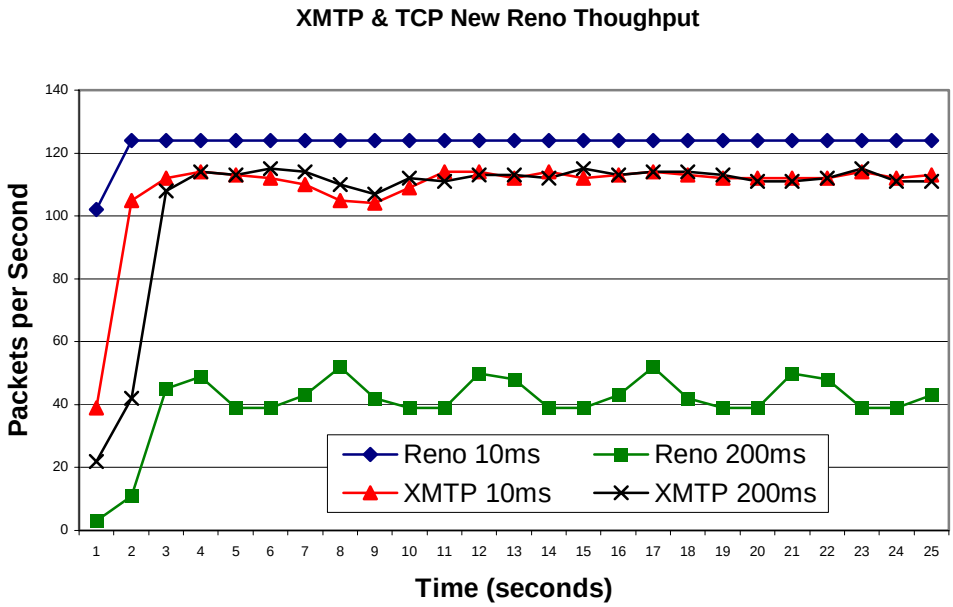



**Figure 9: XMTP and TCP New Reno Throughput**

Figure 9 shows results for XMTP and TCP New Reno for lowest and highest RTTs are compared. XMTP is not dependant on the routers queues to compensate for different bandwidth/delay products, so it can achieve its maximum bandwidth with any delay. On the other hand, it is limited by its synchronization avoidance mechanism to stay below the absolute link maximum, which is achieved by TCP for low delays. By design, XMTP stays below the maximum bandwidth on a path to allow that queues that may form at routers to drain. If data were transmitted at the maximum available bandwidth, it would be impossible for queues to drain, because there would be no bandwidth left over. Although counter-intuitive, this leads to greater link utilization under most scenarios by

avoiding congestion losses. In this scenario, however, this leads to link under-utilization, but in a smaller scale than TCP for most link delays.

### 3.10.2 Convergence to available bandwidth

A non-responsive flow by definition will not change its rate due to congestion. Multimedia flows are often non-responsive either because they lack the mechanism to measure network conditions, or because they cannot change the source encoding rate. In this case, most applications assume that it is better to suffer some losses than to fail in transmission, especially because most multimedia streams will degrade gracefully in the presence of a small percentage of losses. This set of experiments depicts the reaction of an established flow (a flow that was given enough time to converge to the bottleneck bandwidth) to a non-responsive flow.

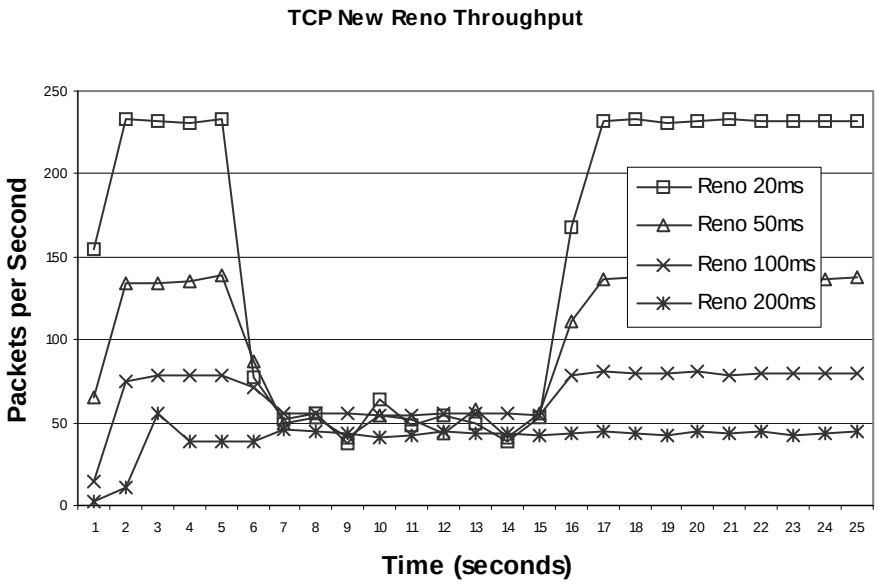



**Figure 10: TCP New Reno response to a CBR flow**

The scenario is the same as the previous experiments. The unresponsive flow uses 1.5Mbps of the 2Mbps bottleneck link. The link delay on the bottleneck link is changed and throughput is measured. The ideal responsive flow should show an inverted square wave. The objective is to measure how long it takes a flow to decrease its throughput, which minimizes losses, and how long it takes to regain its previous throughput once the non-responsive flow is gone.

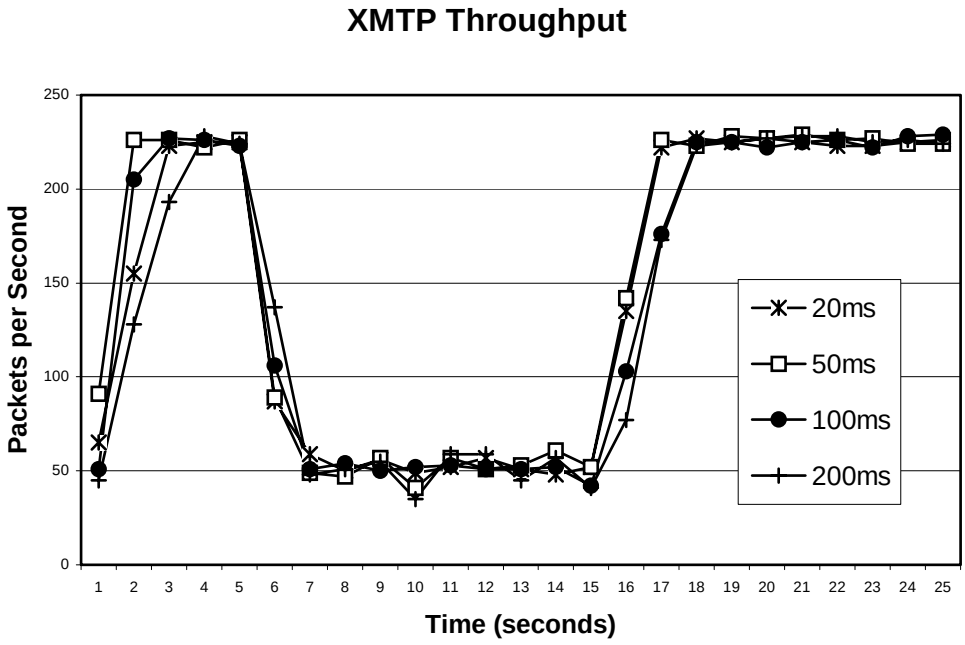



**Figure 11: XMTP Response to a CBR Flow**

In Figure 11 the experiment is repeated for XMTP. It also reacts as expected, dropping its throughput in response to the CBR flow and returning to the same level after the flow is gone. It takes two seconds for XMTP to drop to the new available bandwidth, and two to three seconds to return to the initial level once the unresponsive flow is gone, depending

on the bottleneck link delay. In Figure 12 XMTP and TCP are compared, and it is shown that they have similar behavior. TCP is faster the XMTP to regain bandwidth once the CBR flow stops, but XMTP is less affected by differences in path delay.

In Figure 10 TCP's response to a square wave flow is shown. For higher RTTs TCP should perform worse than XMTP, because it would take it longer to recover from the losses, but in these experiments the effect is masked by the lower overall throughput, and at 200ms TCP's throughput is under the 0.5Mbps left over from the unresponsive flow.

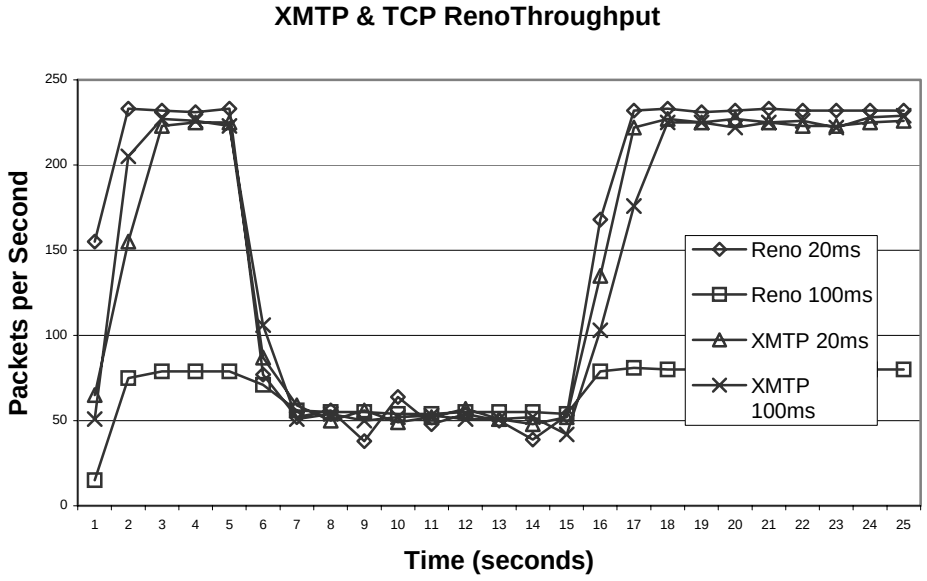



**Figure 12: TCP and XMTP response to a CBR Flow**

### 3.10.3 Fairness and Bandwidth sharing

A congestion control algorithm should be fair in the sense that when multiple flows are present, each gets an equal share of bandwidth. This is hard to achieve, and even TCP will not be fair if conditions are not the same for all flows. If flows with larger RTTs are

sharing the same link, they tend to get a smaller share because they take longer to recover from losses, for example. In this experiment, simulations are run with five flows, changing the mix from 5 TCP flows to 5 XMTP flows. The total number of packets that was received for each flow is recorded. Then the usage, or total number of packets is calculated, along with the fairness. To calculate fairness, we use Jain's fairness index shown in Equation 1.

$$(\sum_i^n x_i)^2 / n(\sum_i^n x_i^2).$$

**Equation 1: Jain's Fairness Index**

The experiment shows XMTP's behavior in presence of reactive traffic. The questions are if XMTP flows will be stable, each flow converging to a fair share of bandwidth without oscillations, and if new flows will be able to get bandwidth from established flows. This is compared with the performance of TCP flows under the same conditions.

A scenario similar to the previous runs was used. Each flow has its own starting and ending node, and share a bottleneck link. There are five flows. The bandwidth on each link is 10Mbps, which gives a share of 2Mbps on the bottleneck link for each of the 5 flows. The delay on the bottleneck link was changed on each run.

The results are for 50ms delay on the bottleneck link. Table 1 shows the results obtained. Each line records an experiment. Darker cells indicate TCP flows. In line 1, the results for the simulation with five TCP flows are shown. Each gets approximately the same number of packets, and the fairness is almost 1. In line 2, one XMTP flows takes place of

one TCP flow. XMTP is less aggressive than TCP at this delay range, so TCP is able to steal bandwidth from XMTP. All TCP flows get more bandwidth they had in the previous experiment. That means that XMTP is not interfering with TCP, and although it is able to use only half of the bandwidth of the other flows, it is not completely stopped by TCP. A better result is obtained on lines 3 and 4, where TCP and XMTP share the link equally. In line 5, because one flow did not perform well, both usage and fairness dropped, but no flow was starved, and TCP performs at the same level it was performing with only TCP flows present. In the last line, we only have XMTP flows. By design, they should be below 10% of link bandwidth, and they are more than that, closer to 17% below. But every flow is getting the same bandwidth, so the fairness is also high.

| | Flow 1 | Flow 2 | Flow 3 | Flow 4 | Flow 5 | Usage | Fairness |
|---|---|---|---|---|---|---|---|
| 5 TCP | 6274 | 6188 | 6188 | 6100 | 6193 | 30943 | 0.999 |
| 1XMTP/4TCP | 2930 | 7137 | 7137 | 6858 | 6866 | 30928 | 0.935 |
| 2XMTP/3TCP | 5480 | 6541 | 6541 | 5998 | 5935 | 30495 | 0.996 |
| 3XMTP/2TCP | 5384 | 5473 | 5473 | 6265 | 5884 | 28479 | 0.997 |
| 4XMTP/1TCP | 2204 | 5712 | 5712 | 6831 | 6192 | 26651 | 0.916 |
| 5 XMTP | 4277 | 5238 | 5238 | 5682 | 5471 | 25906 | 0.990 |

**Table 1: Fairness for XMTP & TCP flows**

In general, because TCP performs better at smaller RTTs, TCP flows get a larger share of bandwidth under these conditions. With larger RTTs, XMTP tends to dominate, although in both small and large RTTs both TCP and XMTP work without starving the competing protocol.

### 3.10.4 Stability

This experiment was designed to test the stability of the congestion control mechanism when multiple flows are sharing the same link. A dumbbell-shaped scenario was created, where 100 nodes are connected to a single node though links with 1Mbps throughput and 10ms delay each. This node is connected to another using a single link with a 50Mbps bandwidth (half the aggregated bandwidth of the links) and variable delay. Finally, the last node has another 100 nodes connected to it. Each node in one extreme is paired with another in the other extreme, and a reliable connection, using either TCP New Reno or XMTP is established. The simulation is run for 25 seconds, and the number of packets received by each connection is recorded. This number is then input into Jain's Fairness Index.

The simulation code is shown below.

```
proc create_topology {} {
global ns n num_node delay

set num_node 100

set aux1 [expr 2 * $num_node]
set n($aux1) [$ns node]

set aux2 [expr  1+ 2 * $num_node]
```

```
set n($aux2) [$ns node]

$ns duplex-link $n($aux1) $n($aux2) [expr 0.5* $num_node]Mb
$delay DropTail
set aux3 [expr 1.5 * 50 * $num_node]
puts $aux3
$ns queue-limit $n($aux1) $n($aux2) $aux3

for {set i 0} { $i < $num_node } { incr i} {
    set n($i) [$ns node]
    set n([expr $i+$num_node]) [$ns node]
    $ns duplex-link $n($i) $n($aux1) 1Mb 10ms DropTail
    $ns duplex-link $n([expr $i + $num_node]) $n($aux2) 1Mb 10ms
DropTail

    }
}

proc finish {} {

global ns delay
$ns flush-trace
exit 0
}

global num_node, n, mix
if {$argc > 1} {
    set mix  [lindex $argv 0]
    set delay [lindex $argv 1]
    set end_time  [lindex $argv 2]
    } else {
          puts "usage: ns dumbell.tcl <mix> <delay> <time>"
          puts " "
          puts "<mix> is 0..100 how many flows are TCP e.g. 100
for all TCP run"
          puts "<delay> is the delay on the bottleneck link e.g.
10ms"
          puts "<time> is how long the simulation will run e.g.
25.0"
          exit 1
          }

set ns [new Simulator]
$ns trace-all [ open out.tr w]

create_topology

# TCP
```

```
for {set i 0} { $i < $mix } { incr i} {
    set tcp($i) [new Agent/TCP/Newreno]
    $tcp($i) set packetSize_ 1000
    $ns attach-agent $n($i) $tcp($i)
    set sink($i) [new Agent/TCPSink]
    set aux3 [expr $i + $num_node]
    $ns attach-agent $n($aux3) $sink($i)
    $ns connect $tcp($i) $sink($i)
    set ftp($i) [new Application/FTP]
    $ftp($i) attach-agent $tcp($i)
    }


# XMTP

for {set i $mix} { $i < $num_node } { incr i} {
    set mmtp($i) [new Agent/mmtp]
    $ns attach-agent $n($i) $mmtp($i)
    set mmtps($i) [new Agent/mmtp]
    set aux3 [expr $i + $num_node]
    $ns attach-agent $n($aux3) $mmtps($i)
    $ns connect $mmtp($i) $mmtps($i)
}

#action
for {set i 0} { $i < $mix } { incr i} {
     $ns at 0.0 "$ftp($i) start"}
for {set i $mix} { $i < $num_node } { incr i} {
     $ns at 0.0 "$mmtps($i) receive"
     $ns at 0.0 "$mmtp($i) start"}

$ns at $end_time "finish"
$ns run
```

**Figure 13: Simulation code**

This simulation is parameterized so the following can be varied: the delay of the bottleneck link, how many of the hundred flows is TCP or XMTP and the duration of the run. Delays of 10ms, 50ms and 100ms were chosen for the bottleneck link, and the mix of TCP and XMTP was varied in 5 steps: 0/100, 25/75, 50/50, 75/25 and 100/0. Then the total number of packets in each flow was measured and the fairness was calculated by

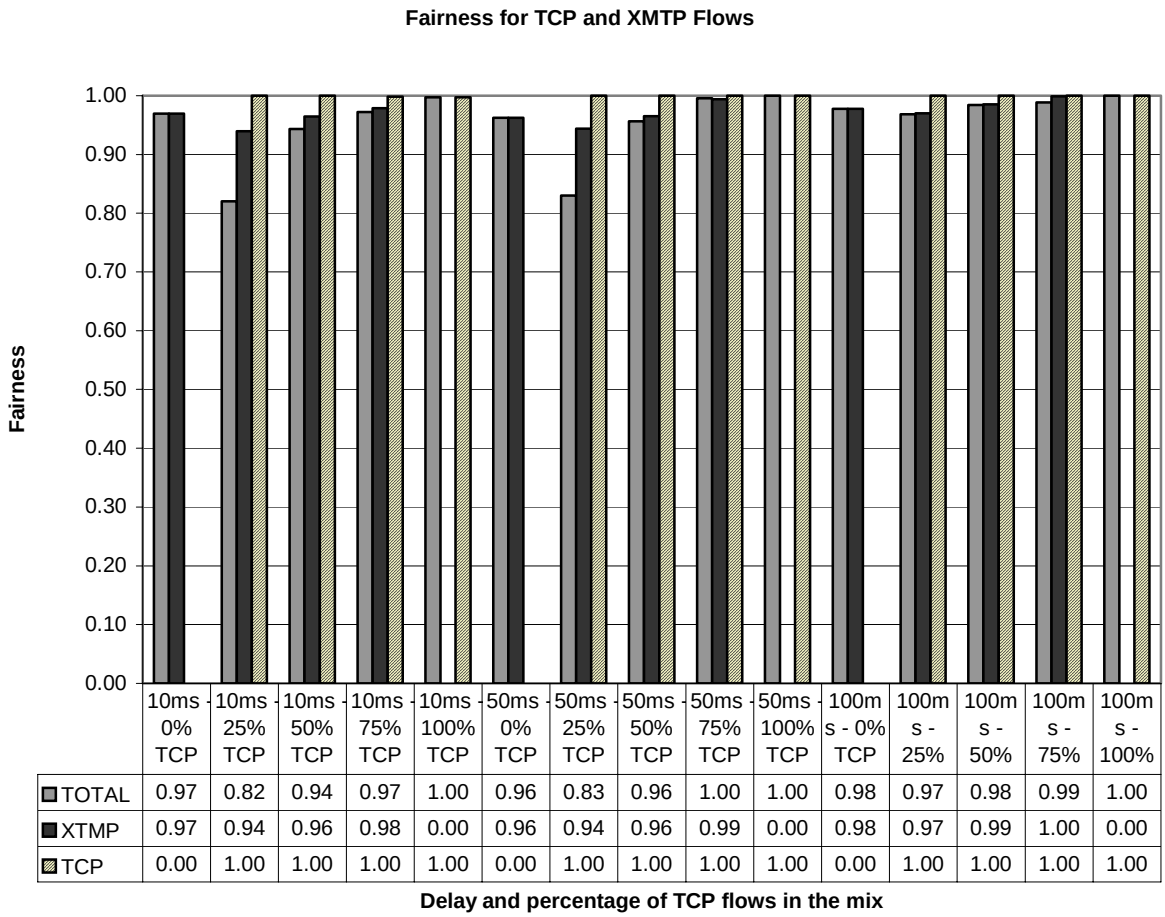
using the following program in awk:

```
BEGIN {
n = 0;
sum = 0;
sumsqr = 0;
}
{
n++;
sum = sum + $1;
sumsqr = sumsqr + $1 * $1;
}
END {
print "number of flows " n
print "total number of packets " sum
fairness = sum * sum / (n * sumsqr);
print "fairness " fairness
}
```

**Figure 14: awk code for calculating fairness**

This results are summarized in Table 2. The column total shows the fairness for aggregated TCP and XMTP flows. The column XMTP shows the fairness within XMTP flows, and the column TCP shows the fairness only among TCP flows.

It can be observed that TCP flows are very fair to each other, as expected, and XMTP flows are also very fair, because fairness never drops below 0.9. But the combined TCP/XMTP flows are not as fair, although the fairness never drops below 0.8. Under the conditions of the runs, TCP was getting more bandwidth than XMTP, and that shows in the fairness. But as the delay got progressive larger, TCP gets less aggressive, and fairness is higher.




| | 10ms 0% TCP | 10ms 25% TCP | 10ms 50% TCP | 10ms 75% TCP | 10ms 100% TCP | 50ms 0% TCP | 50ms 25% TCP | 50ms 50% TCP | 50ms 75% TCP | 50ms 100% TCP | 100ms - 0% TCP | 100ms - 25% | 100ms - 50% | 100ms - 75% | 100ms - 100% |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TOTAL | 0.97 | 0.82 | 0.94 | 0.97 | 1.00 | 0.96 | 0.83 | 0.96 | 1.00 | 1.00 | 0.98 | 0.97 | 0.98 | 0.99 | 1.00 |
| XTMP | 0.97 | 0.94 | 0.96 | 0.98 | 0.00 | 0.96 | 0.94 | 0.96 | 0.99 | 0.00 | 0.98 | 0.97 | 0.99 | 1.00 | 0.00 |
| TCP | 0.00 | 1.00 | 1.00 | 1.00 | 1.00 | 0.00 | 1.00 | 1.00 | 1.00 | 1.00 | 0.00 | 1.00 | 1.00 | 1.00 | 1.00 |

**Table 2: Fairness for TCP and XMTP flows for different bottleneck delays**

### 3.10.5 TCP Friendliness

Fairness is a very close concept to “friendliness”. Both measure the concept of a network “good neighbor”. In the last set of experiments, the objective is to investigate how TCP and XMTP interact. Two flows, one XMTP and one TCP share the same path, and the throughput in packets per second is measured.

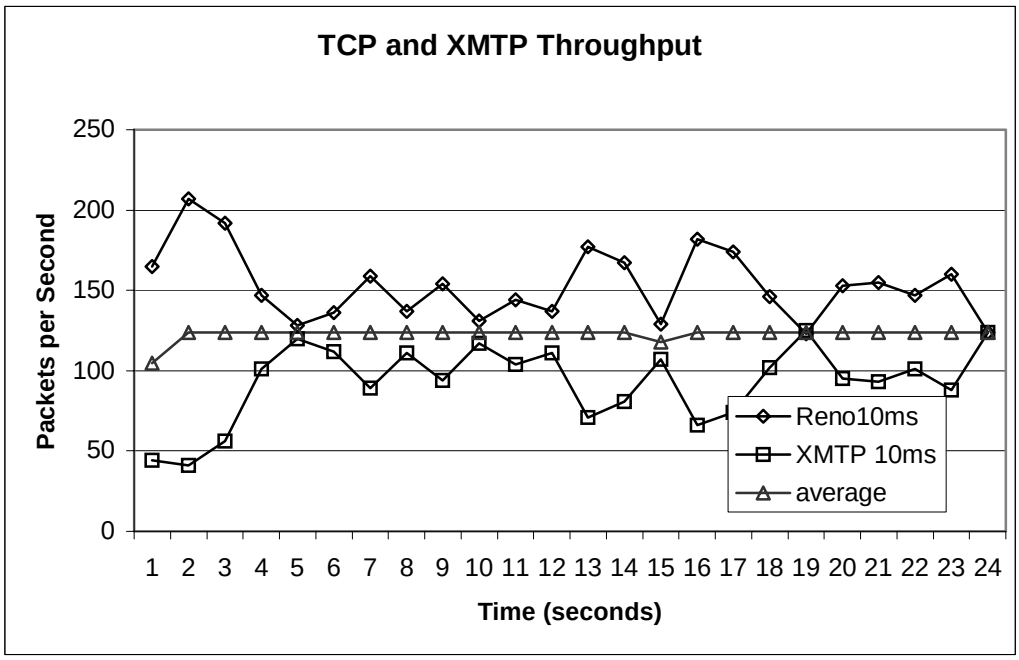



**Figure 15: TCP and XMTP Througput (10ms delay)**

In Figure 15, it is shown how both flows share a link with 10ms delay. With low delay, TCP is more aggressive, and tends to get more bandwidth than XMTP.

The converse is true for larger RTTs. With a 50 ms delay, the graph (Figure 16) shows that XMTP dominates. Nevertheless, in both cases neither TCP nor XMTP starve out the other. Even with larger delays (Figure 17), when TCP cannot achieve full link bandwidth, XMTP does not starve out TCP.

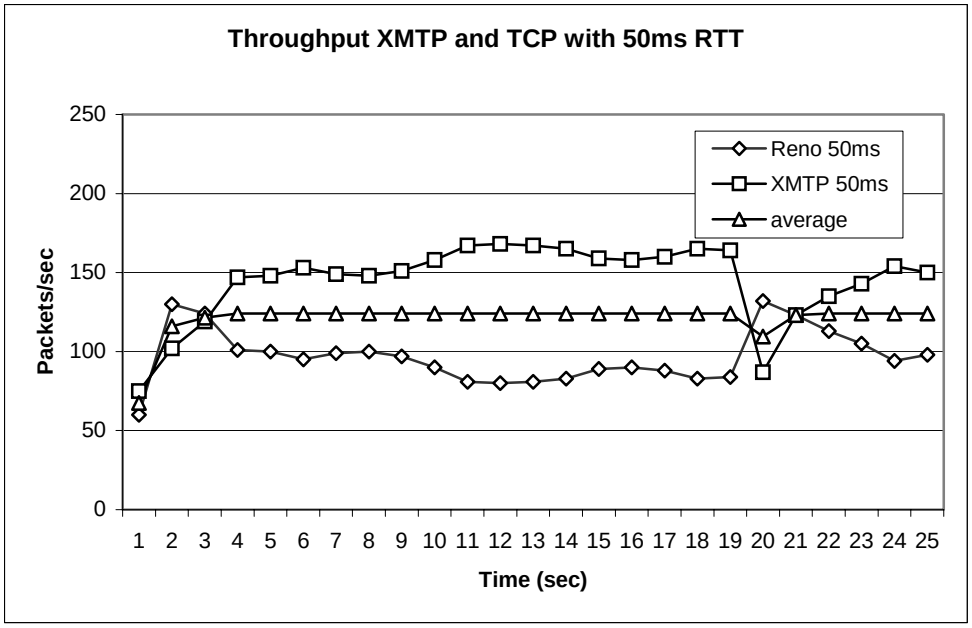



**Figure 16: TCP and XMTP Throughput (50ms delay)**

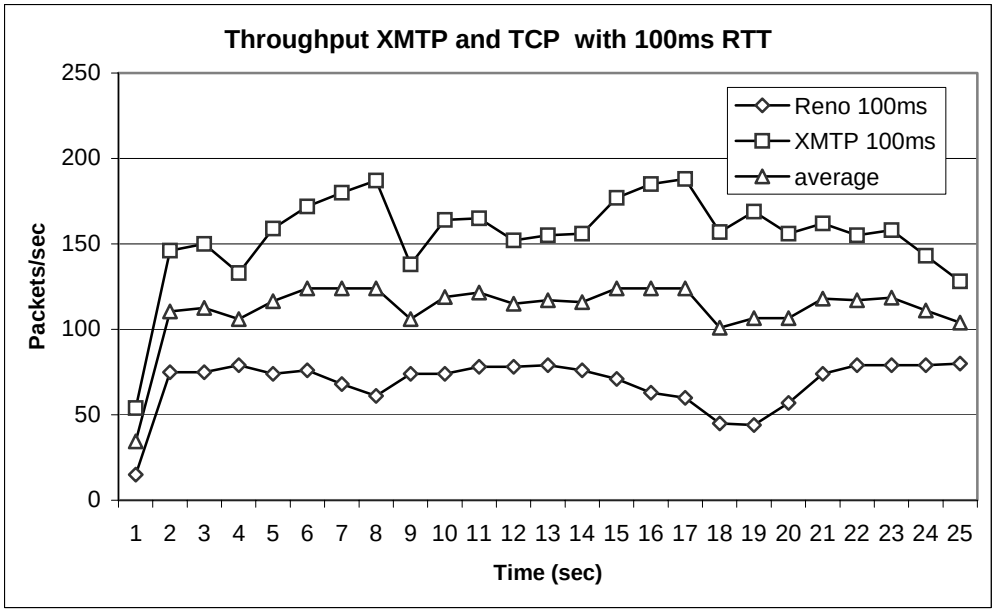



**Figure 17: TCP and XMTP Throughput (100ms delay)**

## 3.11 Conclusions and Future Research

This Chapter presented the Homeostatic Congestion Control algorithm, which is suitable for congestion control of rate-based transport protocols. HCC is a congestion avoidance algorithm, because it can react to changing network conditions before congestion takes place, by measuring available bandwidth and dropping the protocol transmission rate if it exceeds the perceived available bandwidth of the network. HCC measures available bandwidth using two strategies: packet pair and jitter correction. To measure increases in available bandwidth, it uses the packet-pair method. This allows it to converge faster to available bandwidth than an AIMD algorithm, but may lead to an overestimation of available bandwidth. To measure decreases in available bandwidth, and to compensate for the overestimation of available bandwidth by the packet-pair measurement, HCC corrects the rate by using the error information, which is the difference between the expected rate and the inter-arrival times of the packets. By adding this quantity (which is the jitter) to the rate, both congestion avoidance and rate correction (from the overestimation) are achieved. HCC has an additional mechanism if congestion avoidance is not enough and losses still occur. If losses are detected, the rate is halved once for each RTT that contains a loss event. Finally, HCC uses a novel strategy to avoid synchronized losses, which can affect rate-based protocols. A randomizer is used to add up to 10% to the final value calculated for the rate. Because the value is random, different flows will use different values for the rate, which will fluctuate slightly, preventing synchronization.

The main contribution of this Chapter is the description of HCC and its evaluation. HCC

was evaluated using simulation. A protocol module called XMTP was created for ns-2. XMTP was run in different scenarios and compared to TCP New Reno. XMTP converges to link bandwidth and responds well to both CBR and TCP flows, regaining bandwidth when competing flows are shut off. XMTP's main disadvantage is its inability to achieve full link bandwidth when a single flow is run, due to the random value added to the rate, but it was shown to be stable and TCP friendly. When multiple flows are run, XMTP achieves good link utilization and fairness.

HCC can be augmented with a loss discrimination heuristic, which will be shown in the next Chapter. One of the weaknesses of HCC is its need of good timers. User space implementations of HCC, which will be described in Chapters 5 and 6 are limited to the timer resolution offered to user programs (normally 10 milliseconds, maximum 1 millisecond). Future kernel implementations will not have this limitation.

The current version of HCC uses a fixed probing interval. One possible improvement would be to change the probing interval according to the RTT of the path. Further research is needed to see if the greater regularity of the flow would compensate having less information about the network. Another area of future research is to build a transport protocol for high-speed networks using HCC. Although it may be necessary to have hardware support to achieve the timer granularities required for high-speed rate-based protocols, HCC may prove to be a good alternative for high-speed networks, preventing at least one problem of ack-clocked design, which is overflowing the router queues before achieving the maxim bandwidth available in a path.

# Chapter 4 Loss Discrimination

## *4.1 Introduction*

This chapter presents a novel heuristic for discerning if a loss was caused by a transmission error or by congestion on the network [MA003]. The main requirement for such heuristic is that it should not cause congestion by misclassifying congestion losses. A secondary requirement is that the heuristic should classify transmission losses as such, to allow improvements in the response to this type of loss. The proposed heuristic is coupled with the congestion control algorithm presented with the previous Chapter. It makes use of the regularity of traffic generated by the rate-based transmission to make predictions more accurate. Loss discrimination is important to mobile hosts because wireless links are more prone to transmission-related losses than wired links.

Advances in communication technology have increased the reliability of wired networks to the point where transmission losses are rare. As a result, losses in the Internet are generally caused by path overuse. Such specific knowledge about network reliability has enabled TCP [AL099] to be tuned to assume that losses are caused by congestion. The introduction of technologies with higher loss rates breaks this assumption. Some losses can be handled by making the lossy link appear reliable. For example, modems provide link-layer error correction in order to mask packet loss due to transmission errors. Such an approach assumes that the transmission losses can be recovered locally in a fashion that will not adversely affect the end-to-end transmission of data. However, these

approaches do not guarantee that no transmission losses will occur. Therefore, it is necessary to adapt transmission protocols to handle transmission losses.

Transport protocols should react to congestion losses in a fashion that helps alleviate the congestion in the network (e.g. reduce offered load). If congestion losses are treated as transmission losses, the sender will not decrease its offered load and more congestion will build up in the network. In response to transmission losses, the packet should simply be retransmitted. If transmission losses are treated as congestion losses, the sender will unnecessarily reduce its offered load, reducing the throughput of the stream. Hence, the miscategorization of losses can have detrimental effects on the communication flow and on the network. In this Chapter, we describe a loss discrimination heuristic that can be combined with congestion control to allow the sender to react appropriately to loss.

The main challenge to loss discrimination lies in the fact that a communication channel may span both wired and wireless links, introducing both congestion and transmission losses into the channel. Approaches that deal with such heterogeneous packet loss attack the problem at all layers of the protocol stack. Reliable link-layers have been proposed [SI093]. One hundred percent reliability at the link layer can interfere with end-to-end estimates of path round trip time (RTT) [BA095]. If less reliability is provided, the end-to-end mechanisms must still be able to handle transmission losses. Network-layer solutions, such as explicit congestion or loss notification [FL094] [BA098], require changes to the infrastructure. Hybrid approaches, such as SNOOP-TCP [BA096], differentiate the lossy from the more reliable part of the path and try to optimize

transmission across each part separately. Deployment of such approaches is limited by the need for intelligent base stations or agents in the network. We support solutions at the transport layer that do not require changes to the infrastructure. Current end-to-end approaches have been limited by the success with which they can discriminate between congestion and transmission losses.

The main contribution of this chapter is an end-to-end mechanism for loss discrimination. The rate-based transmission approach uses timing information gathered at the receiver to infer the level of congestion on the path between the sender and the receiver. As losses are detected at the receiver, the protocol uses this timing information to discriminate between congestion and transmission losses.

The rest of this Chapter is organized as follows. Section 4.2 discusses the new approach to loss discrimination in the context of current research in the area. Section 4.3 presents the basis for the new loss discrimination heuristic, discussing the characterization of network parameters, specifically congestion. Section 4.4 describes the inter-relationship between congestion control and loss discrimination. Section 4.5 presents an evaluation of the heuristic based on simulation results. Finally, Section 4.6 presents the conclusion and future research directions.

## *4.2 Dealing with Transmission Losses*

Successful loss discrimination is essential for effective communication in environments where losses may be both congestion- and transmission-based. In the case of a congestion

loss, the sender should reduce the amount of traffic that it is putting into the network while in the case of a transmission loss the sender should simply continue transmitting. Without loss discrimination, all losses are attributed to either congestion or transmission. In the first case, transmission losses will be misinterpreted and performance will suffer. In the latter case, congestion losses will be misinterpreted and congestion will increase along the network path.

The problem of adapting transport protocols for networks with heterogeneous loss characteristics has been attacked from different directions. Infrastructure-based solutions try to hide the losses from TCP by adding changes to the intervening path, either at the link-layer or at the transport-layer. A second class of solutions proposes changes to TCP. A third class of solutions proposes new transport protocols instead of changing TCP. In this section, we discuss the approach taken and compare it to existing approaches. A detailed survey of these approaches can be found in [PE000].

### 4.2.1 Infrastructure-Based Approaches

Infrastructure-based approaches are appealing because they require no changes to TCP and because the link layer has direct knowledge of transmission errors. Link-layer error recovery is based on two mechanisms: Automatic Repeat Request (ARQ) and Forward Error Correction (FEC). While both approaches make a link appear reliable, neither is free. FEC imposes overhead on every packet and is computationally expensive. Because ARQ may increase the latency of the link and generate out-of-order packets, it can lead to

worse overall performance for protocols that need reliable RTT estimates and expect packets to arrive in order [BA096]. While link-layer approaches are often appropriate, they do not always recover all losses. Therefore, we believe that even with link-layer recovery, end-to-end approaches will still be necessary. The heuristic described in this Chapter will work correctly with modified link-layers. FEC will transparently introduce relatively fixed overhead per packet, and so will not adversely affect timing estimates. ARQ may skew RTT estimates and so render any end-to-end approach less effective. Additionally, any inclusion of reliability at the link layer may impose high overhead for streams that do not require any reliability

To avoid such overhead for non-TCP traffic, it has been proposed to make the link-layer TCP-aware. An example of this approach is SNOOP_TCP [BA095], which changes the wireless base-stations, allowing them to cache unacknowledged packets. If the base station perceives duplicate acknowledges, it suppresses them, and sends the cached data instead. It also retransmits locally cached packets using timeouts, which should be smaller than the sender's timeout. Although this approach is effective, it requires modifications to all base stations involved in the communication.

Indirect TCP (I-TCP) [BA995] splits the TCP connection into two parts, one over the wired and another over the wireless network, allowing each section to be optimized for the appropriate type of losses. The base-station acts as the TCP receiver and is responsible for forwarding data to the wireless host after it has received it. This violates TCP end-to-end semantics, since data will be acknowledged before being received at the

wireless host. By discriminating between transmission and congestion losses, the proposed heuristic enables successful transmission over channels that experience both types of losses without requiring specialization for either type.

### 4.2.2 Hybrid Approaches

Explicit knowledge of losses can be exposed to the transport-layer from the network and link layer. Explicit Congestion Notification (ECN [FL094] [RA001]) allows the routers to inform TCP senders of incipient congestion, so they can drop their sending rate and avoid congestion. Explicit Loss Notification (ELN [KR004]) allows the base-station, the wireless router, to inform the sender of a transmission loss, so the packet can be retransmitted and no congestion control measures need be applied. Although such explicit information is beneficial, such approaches require changes to both the transport protocol and the infrastructure. By using end-to-end path information, the proposed heuristic strives to infer such explicit information.

### 4.2.3 End-to-End Approaches

Since it is expensive to deploy infrastructure-based approaches, adaptations have been proposed for TCP to help it perform better in environments with higher loss rates. There are two challenges to end-to-end approaches. The first is accurate estimation of available bandwidth and the second is appropriate reaction to loss.

Traditional versions of TCP (i.e. Reno and Tahoe [CO095]), are optimized for minimal

losses and so do not react well to multiple losses in the same window [FA096]. The use of Selective Acknowledgements (SACK [MA096] [FL100]) has been suggested to alleviate some of this problem, but does not address the issue of loss discrimination.

In order to do accurate loss discrimination, an end-to-end protocol needs information about the state of the network that can be estimated through observations of the transmission stream. Ideally, transit time for each packet should be used since this provides information about the network in the direction of the transmission. Solutions such as TCP Vegas [BR095] strive to achieve good estimates by monitoring RTT. The challenge lies in the fact that asymmetry on the path can cause inaccurate estimates of transit time. TCP Eifel [LU000] uses timestamps in each TCP packet to provide accurate estimate of RTT, but does not address the issue of different levels of congestion on the reverse path.

TCP Santa Cruz [PA999] also uses a timestamp returned from the receiver. The goal of TCP Santa Cruz is to estimate the level of queueing in the bottleneck link of a connection and maintain an optimal number of packets in the bottleneck of the connection, without congesting the network. Congestion-control is based on the tracking of network load. The timestamp TCP Santa Cruz uses is very similar to our proposal, although our rate-based approach is simpler and provides more accurate timing information.

Most similar to our heuristic-based approach is TCP-Aware [BI099], which monitors the transmission stream to determine transmission losses. The limitation of this approach is that TCP-Aware requires that the last hop be both the bottleneck and the lossy link. It is

also unclear how accurate the heuristic from TCP-Aware are in the presences of multiple streams.

TCP Pacing [AG000] is another technique used for better bandwidth usage. The central idea is that the burstiness of TCP can lead to overflowing queues even when there is no congestion. The solution would be to spread out the packets, sending them at the same rate they would be sent, but over a period of time and not back-to-back. We advocate the development of rate-based protocols for the same reason. Although some problems with global synchronization have been discussed, we believe that the congestion avoidance techniques presented in this paper can prevent the synchronization of losses that can lead to link underutilization.

Instead of reengineering TCP to deal with lossy environments, new protocols have been proposed. An example is Wireless Transmission Control Protocol (WTCP) [SI099], which was designed to provide a reliable transport protocol for CDPD. WTCP is rate-based, and advocates a split connection model. A proxy ends the TCP connection from the sender and initiates a WTCP connection with the mobile. WTCP uses the same algorithms as TCP for connection management and flow control, but has its own congestion control algorithms. WTCP measures the interarrival time at the receiver and uses this information to set the sending rate, and uses congestion discrimination to deal with wireless losses. WTCP uses heuristics based on detecting congestion, and tagging losses as transmission losses if the network is considered uncongested. The presence of congestion is based on observations about long-term jitter. In Section 4.4, we discuss the

problems with using long-term jitter, and advocate the use of recent history for loss discrimination based on information collected at the receiver.

## *4.3 Network Characterization*

An end-to-end approach to loss discrimination is limited by the information that can be inferred from the behavior of the transmission stream. Therefore, it is necessary to have effective mechanisms for estimating path characteristics, specifically the presence of congestion along the path.

### 4.3.1 Path Characteristics

The basis of this loss discrimination technique is accurate determination of congestion along the path from sender to receiver. In order to monitor congestion, it is necessary to understand the characteristics of the end-to-end channel, specifically the expected amount of bandwidth available to each flow. If multiple flows are present, the available bandwidth should be divided fairly among the flows. In an ideal environment, this can be achieved by the exact knowledge of the available bandwidth. In a dynamic environment, if every flow is trying to achieve maximum bandwidth without causing congestion, dynamic equilibrium can be reached. This can be seen, for example, when multiple TCP flows share the same link.

The challenge lies in the determination of available bandwidth and the translation of this value to the maximum share of bandwidth for a particular flow. If this estimate is too

low, the flow will not receive its share of bandwidth and so its throughput will be reduced. On the other hand, overestimates may cause congestion along the path. TCP flows determine this maximum by pushing the limits of the network. Once the limit has been reached, congestion will occur, causing loss in the TCP stream, and TCP will reduce the amount of bandwidth it is using.

In comparison, the central idea of the heuristic is to monitor path characteristics in order to determine available bandwidth without causing congestion. Initial estimates of path bandwidth are measured using the packet pair method [KE092]. Two packets are sent back-to-back and their interarrival time is measured. Since the packets were sent back-to-back, the timing of the arrivals represents the current limit of the network. Packets should thus be sent separated by this time period to achieve maximum bandwidth usage and avoid causing congestion in the network. Since available bandwidth is a moving target, this technique can be used periodically throughout the life of the transmission stream to probe the path for up-to-date estimates of bandwidth. Since this approach strives to avoid causing congestion, the next challenge is how to use implicit information about the characteristics of the path to infer that there is congestion building in the network.

### 4.3.2 Congestion

In end-to-end communication, the receiver is in the best position to collect information about the communication channel. The receiver can distill this information and send the results back to the sender to affect changes in the transmission stream. For rate-based

transmission, the receiver expects to receive packets at regular intervals, as determined by the sending rate. Information about the channel can be inferred from the difference between the expected and actual arrival time of packets.

Information at the receiver is based on both observation and protocol parameters. The main piece of observed data is the arrival time of a packet. Let the arrival time of packet *i* be $AT_i$. The arrival time of a packet, as shown in Equation (1), is the sum of the time it was sent, $TS_i$, and the time it was traveling through the network, $TT_i$.

$$AT_i = TT_i + TS_i \qquad (1)$$

In order for this information to be useful to the receiver, the receiver must have some knowledge about $TS_i$. If a rate-based protocol is being used, the time between sending packets at the sender is a fixed period, P. The send time of a packet is thus given by the send time of the previous packet plus the period:

$$TS_{i+1} = TS_i + P \qquad (2)$$

Since packets are traveling through a dynamic network, the travel time for packets will vary over time. We can divide the travel time in two components, the flight time ($FT_i$), which is the sum of the propagation times from router to router, and the queue time ($Q_i$), which is the total time the packet is queued inside the routers on the path.

$$TT_i = FT_i + Q_i \qquad (3)$$

If routes are not changing and packets are the same size, the flight time is constant,

because FT only depends on the transmission speed of the media used for transport. The queue time has three components, the time it takes to process a packet and put into the appropriate outbound interface queue, the time the packet waits in the queue, and the time it takes to dequeue a packet and send it. The first and last component should be constant or linear based on packet size.

The number of packets in queues along the path of the communication represents the load in the network. As this number increases, a packet in the transmission stream will experience longer wait times at the routers, and so longer end-to-end transmission times. On the other hand, as network load decreases, end-to-end transmission time will decrease down to the limits of the transmission medium. In a dynamic network dominated by bursty TCP traffic, variations on the network load (i.e. the number of packets in the queues along the path) will be observed by changes in end-to-end transmission time.

The relationship between arrival time for successive packets can indicate information about the state of the network. The interarrival time of two packets, $IAT_{i, i-1}$, is defined as the difference of their arrival times, or

$$IAT_{i, i-1} = TT_i - TT_{i-1} + TS_i - TS_{i-1} \qquad (4)$$

From Equation (2), we can reduce Equation (4) to:

$$IAT_{i, i-1} = P + TT_i - TT_{i-1} \qquad (5)$$

The load in the network can be inferred by comparing the expected arrival time of a

packet, $E_i$, to the actual arrival time, $AT_i$, This difference is the jitter for packet *i*, $J_i$.

$$J_i = AT_i - E_i \quad (6)$$

If all information is available at the receiver, expected packet arrival times should all be based on the sending time of the first packet, $AT_1$, and a multiple of the period, $i * P$. Since this is a distributed algorithm, it is complicated for the receiver to determine a value for $AT_1$. Therefore, we calculate the expected arrival time for a packet based on the arrival of the previous packet, $TT_{i-1}$ (Equation 7).

$$E_i = AT_{i-1} + P \quad (7)$$

Hence, we can calculate jitter at packet *i* to be the difference between the interarrival time and the period:

$$J_i = AT_i - (AT_{i-1} + P) \quad (8a)$$

$$J_i = IAT_{i,\,i-1} - P \quad (8b)$$

$$J_i = TT_i - TT_{i-1} \quad (8c)$$

Therefore, the jitter is governed by the difference in the packets' travel time. The only variant, then, is the time waiting at the queue. If we combine Equations (8c) and (3) along with the assumption that $FT_i$ is a constant, we have:

$$J_i = FT_i - FT_{i-1} + Q_i - Q_{i-1} \quad (9a)$$

$$J_i = Q_i - Q_{i-1} \quad (9b)$$

Therefore, the main component of the jitter is the time the packets wait in the routers' queues.

If the service rate is greater than the arrival rate at any router, the queue size should be close to zero, growing only because of burstiness in traffic. Since the definition of expected arrival time is based on the actual arrival time of the previous packet, negative jitter can occur when the first packet experiences longer queuing delays than the second packet. If the service rate at any queue along the path is smaller than the arrival rate, congestion will occur. As the queue grows, the jitter will be positive. At some time the queue will overflow, and packets will be dropped. Even in a stable network queues will grow and shrink. Therefore, the jitter values will be positive and negative over time. In a stable environment, the sum of the jitters should stay near zero.

Minimum transmission time occurs when the network is idle (no packets enqueued along the path), and maximum transmission time occurs when the queues at every router are full. The effect of dropped packets is noticeable by the reduced waiting time of succeeding packets that were successfully queued. Consider the queue in Figure 18. As packet 8 arrives, there is room in the queue, and the packet is enqueued immediately after packet 7. The same is true for packet 9, and it is enqueued immediately after packet 8. As the packets are processed, the spacing between these packets remains the same. In Figure 19, which illustrates a drop-tail queuing strategy, the arrival of packet 8 occurs when the queue is full and the packet is dropped. Before packet 9 arrives, packet 1 is processed and

space is made, and packet 9 is enqueued immediately after packet 7. This allows packet 9 to be processed one time slot earlier than expected. Given no other timing changes in the transmission of the packet, packet 9 will arrive earlier than expected at the receiver.

If the dropping strategy affects packets within the queue, the net effect is the same. For example, if packet 8 has a higher priority than packet 4, packet 4 may be dropped out of the queue (see Figure 20). If all of the odd packets belong to the same stream, the arrival of packet 5 will be earlier than expected and the receiver doesn't have to wait for packet 9 to determine that there was a congestion loss. Essentially, the effects of the loss will be perceived at the arrival of the next packet in the queue after the dropped packet. In this way, a flow will notice the effect of dropped packets from other flows.

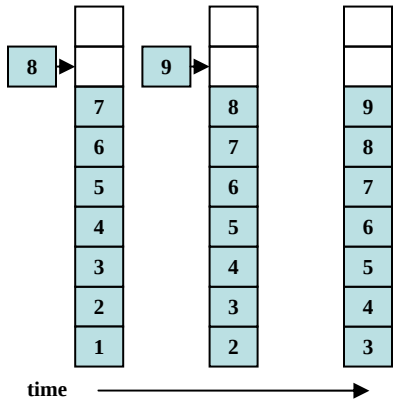



**Figure 18: Successful Queueing**

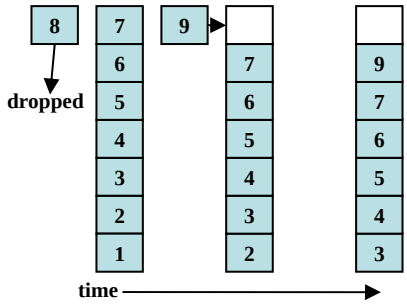



**Figure 19: Dropped Packet at Queue**

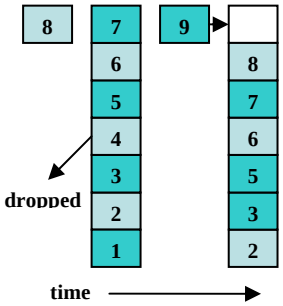



**Figure 20: Dropped Packet with Priority**

## *4.4 Congestion Avoidance and Loss Discrimination*

Without explicit loss or congestion notification, successful loss discrimination is dependent on implicit mechanisms. These mechanisms can be geared toward identifying congestion losses or identifying transmission losses. An end-to-end approach is limited to information at the transport layer, and so completely accurate transmission loss determination is not possible. On the other hand, identifying congestion losses is tightly coupled to determination of congestion in the network. Accurate determination of

network state is complex and often not feasible. To this end, the loss discrimination mechanism is based on a heuristic that integrate congestion avoidance techniques. By monitoring end-to-end channel characteristics, the state of the path between the sender and receiver can be estimated. As a loss is observed, this information is used to determine if the cause of the loss was from congestion along the path. If no congestion is indicated, the loss is determined to be transmission-based. In effect, the result of the loss discrimination is fed into the congestion control mechanisms to determine how to react.

### 4.4.1 Ideal Loss Discrimination

In an idealized network, congestion losses only occur when the bottleneck link has reached saturation. Therefore, any loss occurring when the saturation level has not been reached is a transmission error. As congestion causes queues along the path to fill, travel time for successive packets will increase. Intuition tells us that it should be useful to consider this increase in order to determine the cause of a packet loss. Unfortunately, the complexity of finding this saturation point hampers the effectiveness of this information.

In order to demonstrate this, consider the long-term cumulative jitter. We define the long-term cumulative jitter of a flow at any time $t$, $LCJ_t$, to be the sum of the jitters of each packet arriving before time $t$.

$$LCJ_t = \sum_{AT_i<t} J_i \qquad (10)$$

Now reducing Equation 10 substituting the definition of jitter from Equation 9, we obtain:

$$LCJ_t = \sum_{AT_i<t} (Q_i - Q_{i-1}) \qquad (11)$$

Simplification of the sum in Equation 11 gives us:

$$LCJ_t = Q_L - Q_1 \qquad (12)$$

Where $Q_L$ is the queueing time of the last packet. The long-term cumulative jitter is merely a mask for the queueing time of the last packet. Therefore, any attempt to guess the saturation point of the network based on long-term jitter is equivalent to guessing the queue space remaining in the bottleneck queue along the path. Since long-term jitter is dependent on $Q_1$, it is relative to the amount of packets already queued along the route when the flow begins. Therefore, long-term jitter is always relative to the state of the network at the beginning of the flow. It is unclear how heuristics that uses the remaining queue space as an indicator of congestion will be able to deal with the situation when the congestion level goes below the initial congestion level. It also important to note that there are potentially many routers along a path that could affect the correct threshold estimation.

Inaccurate estimates of remaining queue space will have the following effect. A high

estimate causes some losses that are due to congestion to be judged transmission errors. The protocol would then not react to the congestion in the network, worsening the problem. If on the other hand, low estimate could cause losses that are actually due to transmission error to be judged congestion errors, causing the protocol to reduce transmission rate unnecessarily. Because of these inherent difficulties the effects of inaccurate estimates, the heuristic developed to discriminate between congestion and transmission losses is not based on long-term jitter.

### 4.4.2 Congestion Avoidance Heuristic

Information about congestion along the transmission path allows congestion avoidance even when no losses are detected in the transmission stream. If a trend signaling high network utilization is detected, the protocol should slow down. By tracking consecutive interarrival times of packets, the receiver is in fact tracking the recent history of network load. By monitoring jitter for consecutive packets, the receiver can make the following observations:

1. An increasing trend in interarrival times of packets signals increased network load
2. A decreasing trend in interarrival time of packets may signal reduced network load or congestion

To address the first observation, it is useful to monitor the consecutive number of arrivals that experience positive jitter. Positive jitter occurs when the packet arrives after its expected arrival time. Consecutive packets experiencing positive jitter indicate increasing

queue sizes and potential congestion. In response, the sender should reduce its offered load to avoid adding to the building congestion in the network.

The second observation is a bit more complicated. A trend of negative jitter may indicate contradictory situations. Reduced interarrival time may be caused by reduced network load. The determination of reduced load is left to bandwidth estimation mechanisms. Therefore, the congestion control algorithm need not respond to negative jitter in this first case. Unfortunately, negative jitter may also indicate that the network was very overloaded. Each instance of negative jitter could indicate a congestion loss and sustained negative jitter could indicate that many packets were dropped, allowing successive packets to arrive early (i.e. experience negative jitter). The second case for negative jitter is caused by congestion, but it is difficult to differentiate it from the first case. Therefore, we leave congestion determination in the face of negative jitter to our loss discrimination heuristic. It is interesting to note that in the presence of routers implementing techniques to control congestion like RED [FL093] [BR098], a receiver will observe negative jitter when a packet from a different flow is dropped. In this case, the loss will look like a transmission and not a congestion loss, as is indicated by our heuristic. If a packet from the original flow is dropped, the response is handled by our loss discrimination heuristic.

### 4.4.3 Loss Discrimination Heuristic

The use of a loss as an indication of congestion is a very powerful tool. The goal is to make sure that tool is used accurately. The proposed loss discrimination heuristic strives

to determine how to react appropriately to losses. Consider two scenarios when losses occur: loss in the presence of positive jitter and loss in the presence of negative jitter.

Increasing jitter indicates an increased load on the network. Congestion losses will not occur until a queue along the way is full. Therefore, a loss during a period of increasing jitter can be considered a transmission loss. It is important to note that even if the loss were actually a misinterpreted congestion loss, the consecutive observations of positive jitter still indicate congestion to the congestion avoidance heuristic. In this case, the congestion avoidance heuristic will react to the indications of congestion, making it unnecessary for the loss discrimination heuristic to react to the loss.

Negative jitter can indicate congestion losses in the queues along the path or the unloading of the network. To aggressively react to the first situation, a loss followed by negative jitter will always be characterized as a congestion loss. This characterization is conservative since the sender will react to a transmission loss during a reduction in network load as if it were congestion, reducing its sending rate. If the network load is reducing, the probing mechanism will be able to recover from such misinterpreted losses.

It is possible that a congestion loss, which would normally be followed by the observation of negative jitter at the receiver, will actually be followed by the observation of positive jitter. This can occur when the succeeding packet experiences delay in a router between the router at which the congestion loss occurred and the receiver. It is expected that these scenarios will be self-regulating – after all, there is a maximum queue size on every router, and even if one router had growing queues, it will eventually reach a

maximum size, when the masking is no longer possible and losses will cause negative jitter.

## *4.5 Evaluation*

The effectiveness of our loss discrimination techniques is based on the accuracy of the discrimination heuristic. The misinterpretation of a loss can adversely affect the throughput of the stream or increase the amount of congestion in the network. Therefore, we must evaluate the probability of each type of misinterpretation. The goal of our initial simulations is to determine an upper bound on the number of such misinterpretations, and will be done by using a CBR flow to simulate XMTP. The last simulations are intended to compare the performance of XMTP to another protocol that does loss discrimination. TCP Westwood ([WA005] [YA004] [GE004]) was chosen because it implements a heuristic for loss discrimination in wireless environments, and TCP Reno was used as a baseline measurement.

### 4.5.1 Evaluation using CBR

A rate-based protocol transporting bulk data can be modeled as a sequence of CBR streams, with data rate varying according to the packet size and the sending period. In the first series of simulations, ns-2 was used to simulate a CBR stream following a path that had competing TCP and CBR flows. The TCP flows cause some links to become congested and lead to dropped packets. In this environment, the heuristic was applied to the trace to see with what confidence the cause of a packet loss could be identified. It

should be clear that by using CBR flows in the simulation the flows do not back off when congestion occurs, which must be part of any congestion control algorithm. The second simulation shows that there is a different behavior when congestion avoidance is used (with a smaller network loading due to congestion control on the flow).

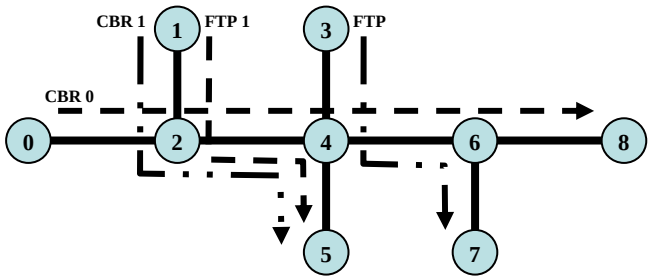



**Figure 21: Simulation Network Topology**

From the simulation the packet number and the arrival time at the destination for each packet are obtained. Losses are indicated by gaps in the sequence numbers. Along with the arrival time, the current rate or sending period is needed for jitter calculations. In this simulation, the period is fixed throughout the experiment. The simulations are set up with the network topology shown in Figure 21. Cross-traffic is indicated by the FTP0, FTP1 and CBR1 flows. The flow being evaluated is CBR0. All losses observed in the simulation are congestion losses. In a sample run, from the ~120,000 packets sent, 8253 gaps due to congestion were detected in the transmission stream.

In order to determine the worst-case effect of the heuristic, transmission losses are not simulated, but instead the effect that a transmission loss that occurred prior to the arrival of each packet is calculated. We assume that the transmission loss would leave all other

timing aspects unchanged. This is a reasonable assumption if the last link is the link subject to transmission losses and this link is not congested. It should be noted, however, that the results will not mirror that of an actual run, but instead give a worst-case result. The reason is simple: if we assume, for example, single losses, both the previous and the following packet have to arrive. In our case, we calculate the timing of single losses for every packet, not every other packet. We can do that because we have the timing information for the packets that have arrived, and we can consider a scenario where, for each packet, their direct predecessor and successor have arrived, but the packet itself has been lost. Therefore, the results will be worse than a real-life scenario, but allow us to fully test our heuristic.

There are two interesting two performance measurements: how many real congestion losses will be detected as transmission losses, and how many transmission losses will be considered congestion losses. If too many congestion losses are considered transmission losses, the use of the heuristic can increase congestion. If transmission losses are considered congestion losses, the performance of protocols that use this heuristic will be negatively affected.

Based on the simulation for single transmission losses, the change in timing can mask up to 26% of congestion losses, as seen in Table 3. The results are similar for two and three consecutive losses. As the number of consecutive transmission losses grows, it becomes harder to discriminate congestion from transmission error. The loss of timing information masks the indicators of congestion. This may point to a more advanced heuristic where it

becomes more conservative as the number of consecutive errors grows. The heuristic is evaluated over varying queue sizes. It correctly identified between 65 and 96% of the congestion losses.

| Queue Size | Consecutive Losses | Congestion Detected | Mistaken Congestion Losses | Actual Congest | Loss Detection % |
|---|---|---|---|---|---|
| 5 | 1 | 6670 | 1583 | 8253 | 0.808191 |
| 10 | 1 | 5754 | 3084 | 8838 | 0.651052 |
| 15 | 1 | 7503 | 85 | 7588 | 0.988798 |
| 25 | 1 | 7673 | 624 | 8297 | 0.924792 |
| 40 | 1 | 7348 | 522 | 7870 | 0.933672 |
| 5 | 2 | 12285 | 4169 | 16454 | 0.746627 |
| 10 | 2 | 13903 | 3570 | 17473 | 0.795685 |
| 15 | 2 | 10137 | 5035 | 15172 | 0.668139 |
| 25 | 2 | 13347 | 3034 | 16381 | 0.814785 |
| 40 | 2 | 14846 | 803 | 15649 | 0.948687 |
| 5 | 3 | 23721 | 929 | 24650 | 0.962312 |
| 10 | 3 | 23262 | 2337 | 25599 | 0.908707 |
| 15 | 3 | 22594 | 157 | 22751 | 0.993099 |
| 25 | 3 | 22673 | 1453 | 24126 | 0.939775 |
| 40 | 3 | 22565 | 715 | 23280 | 0.969287 |

**Table 3: CBR-Based Congestion Classification**

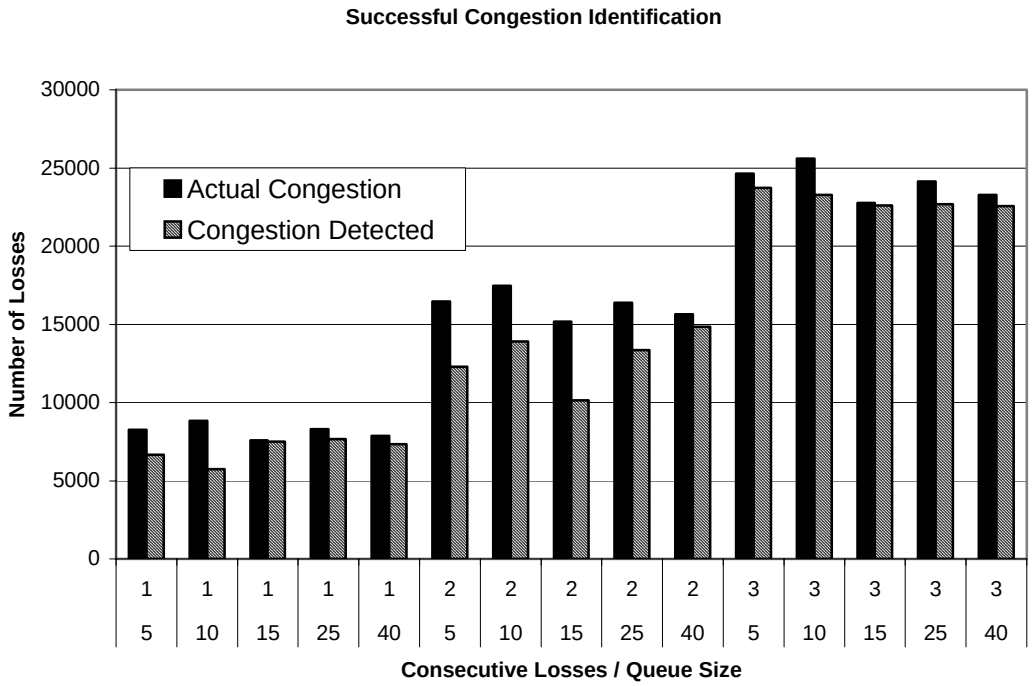



**Figure 22: Acuity of congestion identification for different queue sizes and number of lost packets**

Figure 22 shows the acuity of the congestion detection given different queue sizes and the number of consecutive packets that are lost. The data was taken from Table 3. Not all congestion losses are being identified, which can cause congestion because the protocol will not back off on the misidentified losses. However, because there is a secondary mechanism to correct the misidentification, and a high percentage of congestion losses is being identified as such, the heuristic can be used successfully.

Table 4 shows the corresponding results for transmission errors. Since the simulation did not have transmission errors, this is simply an evaluation of the situation had each packet actually been lost. For transmission losses, this is a worst cases analysis. These results are

used to help evaluate how well the heuristic performs.

| Queue Size | Consecutive Losses | Transmission Losses Detected | Mistaken Transmission Losses |
|---|---|---|---|
| 5 | 1 | 70848 | 1583 |
| 10 | 1 | 70349 | 3084 |
| 15 | 1 | 74418 | 85 |
| 25 | 1 | 74034 | 624 |
| 40 | 1 | 75297 | 522 |
| 5 | 2 | 70762 | 4169 |
| 10 | 2 | 68277 | 3570 |
| 15 | 2 | 69234 | 5035 |
| 25 | 2 | 69070 | 3034 |
| 40 | 2 | 73045 | 803 |
| 5 | 3 | 58725 | 929 |
| 10 | 3 | 57643 | 2337 |
| 15 | 3 | 60420 | 157 |
| 25 | 3 | 59684 | 1453 |
| 40 | 3 | 61327 | 715 |

**Table 4: CBR-Based Transmission Loss Classification**

Figure 23 shows the number of transmission losses detected, and the number of transmission losses that were not detected. The number of losses not detected is very low, which shows that the heuristic is successful in detecting transmission losses.

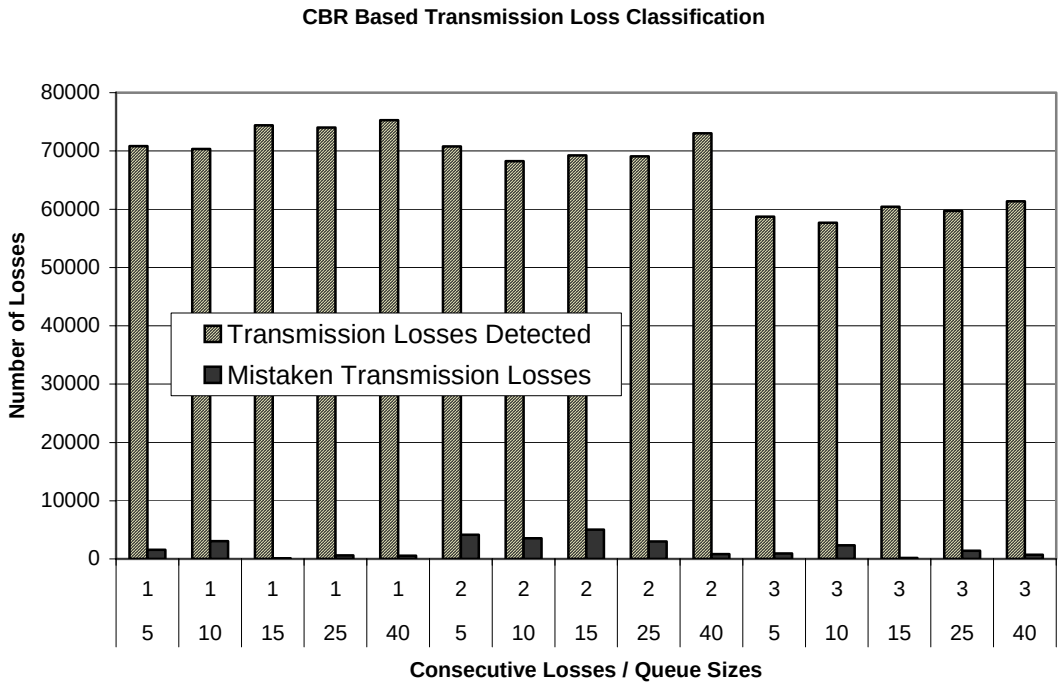



**Figure 23: Acuity of transmission loss identification**

The second set of simulations uses a version of XMTP, described in the previous Chapter, augmented with the loss discrimination heuristic. The goal of this simulation is to show that the number of congestion losses can be significantly reduced during transmission by adhering to the homeostatic congestion control and the proposed loss heuristic. Again, the simulation is run using the network topology in Figure 21. XMTP is run between the same nodes used previously by the CBR streams. There is a TCP stream running across the bottleneck at node 2. Transmission losses are absent in this simulation, and the same calculations are used to obtain the worst-case findings, as was done in the CBR simulations.

| Queue Size | Consecutive Losses | Transmission Losses Detected | Mistaken Transmission Losses |
|---|---|---|---|
| 5 | 1 | 25836 | 147 |
| 10 | 1 | 30175 | 153 |
| 15 | 1 | 31989 | 163 |
| 25 | 1 | 31393 | 163 |
| 40 | 1 | 36082 | 202 |
| 5 | 2 | 26660 | 318 |
| 10 | 2 | 31294 | 345 |
| 15 | 2 | 32955 | 356 |
| 25 | 2 | 32742 | 359 |
| 40 | 2 | 37647 | 410 |
| 5 | 3 | 27527 | 501 |
| 10 | 3 | 32248 | 549 |
| 15 | 3 | 34037 | 562 |
| 25 | 3 | 34251 | 559 |
| 40 | 3 | 39049 | 627 |

**Table 5: Protocol-Based Congestion Loss Classification**

As is shown in Table 5, the number of losses due to congestion is dramatically reduced by reacting to the congestion indicators. It can also be seen that the heuristic is still accurately discriminating between congestion and transmission losses, even though now the stream is not a single CBR stream but is adapting to changes in available bandwidth

and has intervening traffic. Similar results are shown for transmission loss in Table 6.

| Queue Size | Consecutive Losses | Congestion Detected | Mistaken Congestion Losses | Actual Congestion | % Loss Detection |
|---|---|---|---|---|---|
| 5 | 1 | 421 | 147 | 568 | 0.741197 |
| 10 | 1 | 466 | 153 | 619 | 0.752827 |
| 15 | 1 | 496 | 163 | 659 | 0.752656 |
| 25 | 1 | 443 | 163 | 606 | 0.731023 |
| 40 | 1 | 511 | 202 | 713 | 0.716690 |
| 5 | 2 | 783 | 318 | 1101 | 0.711172 |
| 10 | 2 | 846 | 345 | 1191 | 0.710327 |
| 15 | 2 | 893 | 356 | 1249 | 0.714972 |
| 25 | 2 | 805 | 359 | 1164 | 0.691581 |
| 40 | 2 | 948 | 410 | 1358 | 0.698085 |
| 5 | 3 | 1095 | 501 | 1596 | 0.686090 |
| 10 | 3 | 1186 | 549 | 1735 | 0.683573 |
| 15 | 3 | 1238 | 562 | 1800 | 0.687778 |
| 25 | 3 | 1118 | 559 | 1677 | 0.666667 |
| 40 | 3 | 1319 | 627 | 1946 | 0.677801 |

**Table 6: Protocol-Based Transmission Loss Classification**

In Figure 24 the number of congestion losses detected is shown with the actual number of congestion losses. The values are taken from Table 5. Because the protocol is adapting to available bandwidth, not as many losses occur. Not all the losses that result from congestion are being identified as such. However, the congestion avoidance characteristic of the protocol compensates for the mistaken identification, and prevents a congestion collapse.

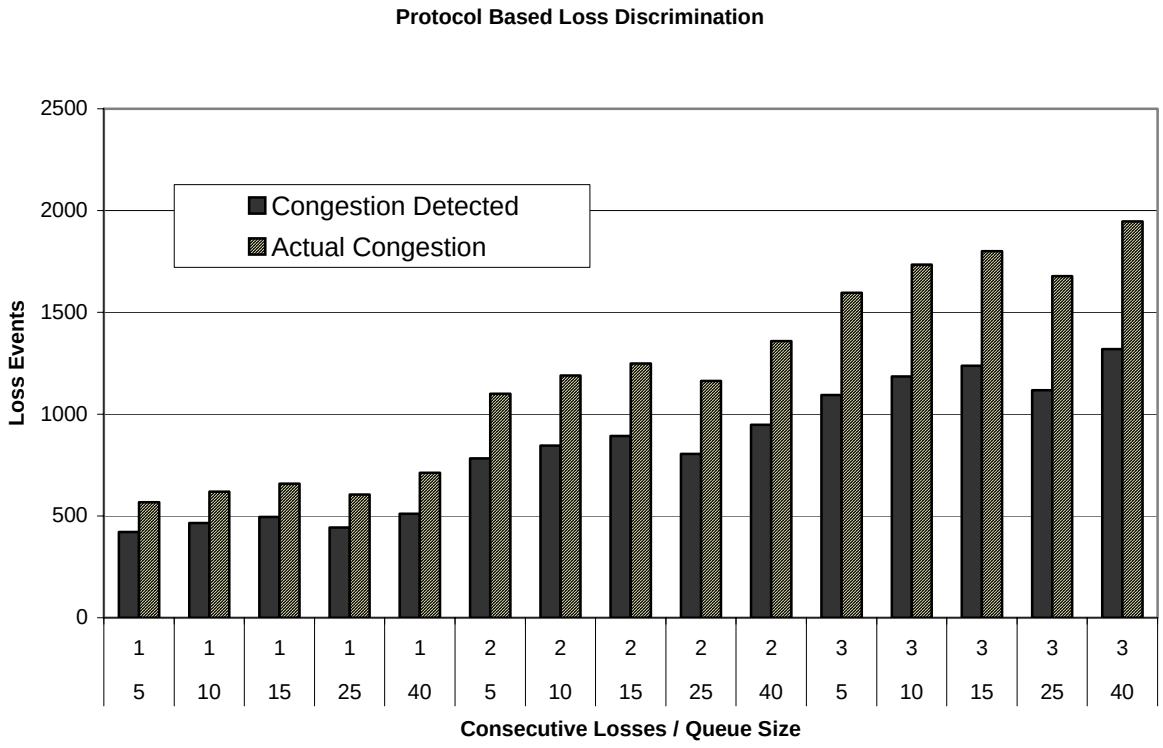



**Figure 24: Acuity of transmission loss detection using XMTP**

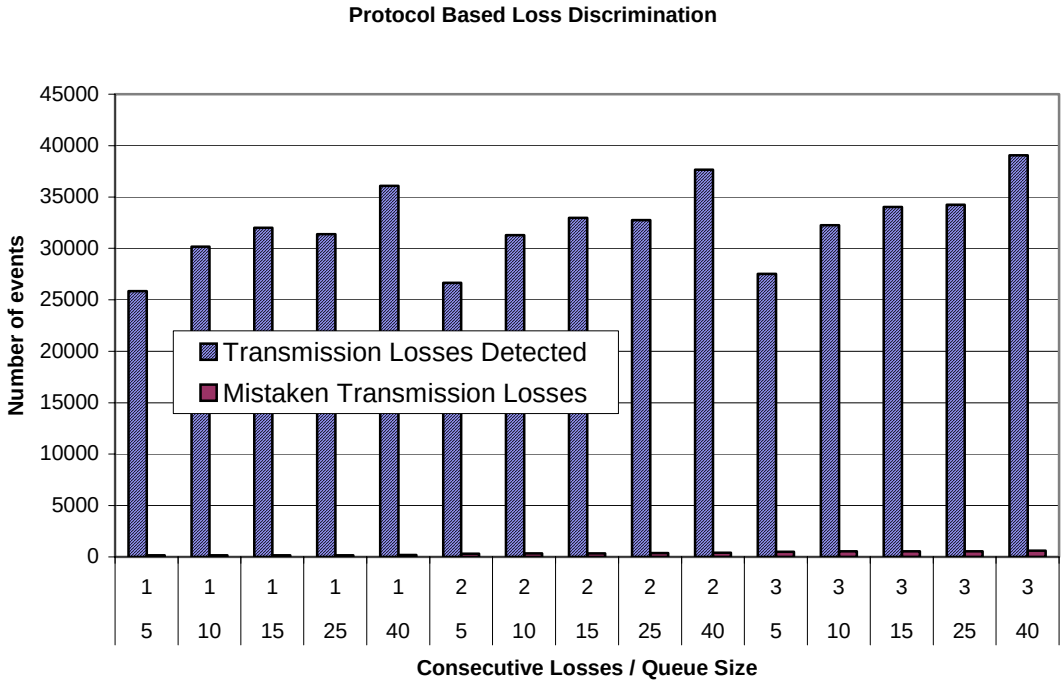



**Figure 25: Acuity of congestion loss detection using XMTP**

### 4.5.2 Evaluation using TCP Westwood and XMTP

In the first simulation, three consecutive 1Mbps links were used, each with a 40ms delay. The throughput is measured by recording the packets arrival time. Each protocol is run independently to see how it would perform with no competing traffic to skew its loss discrimination heuristic.

The first simulation is the baseline for the three protocols, the performance with no loss. XMTP has the best result because it doesn't cause congestion like TCP, although the slope is not as steep as TCP when its congestion window is set.

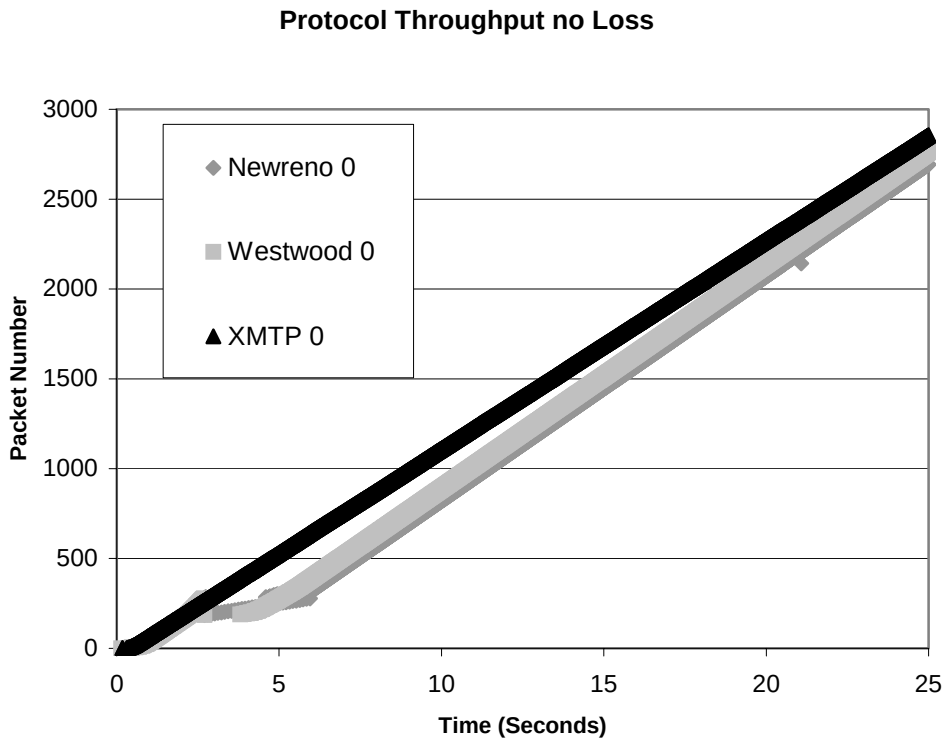



**Figure 26: Protocol throughput no loss**

The second simulation shows the throughput with 0.1 % loss. The results are similar to those in the first run. XMTP has a slight performance hit, and both TCPs show a good slope after the first congestion sets their congestion window with the right value.

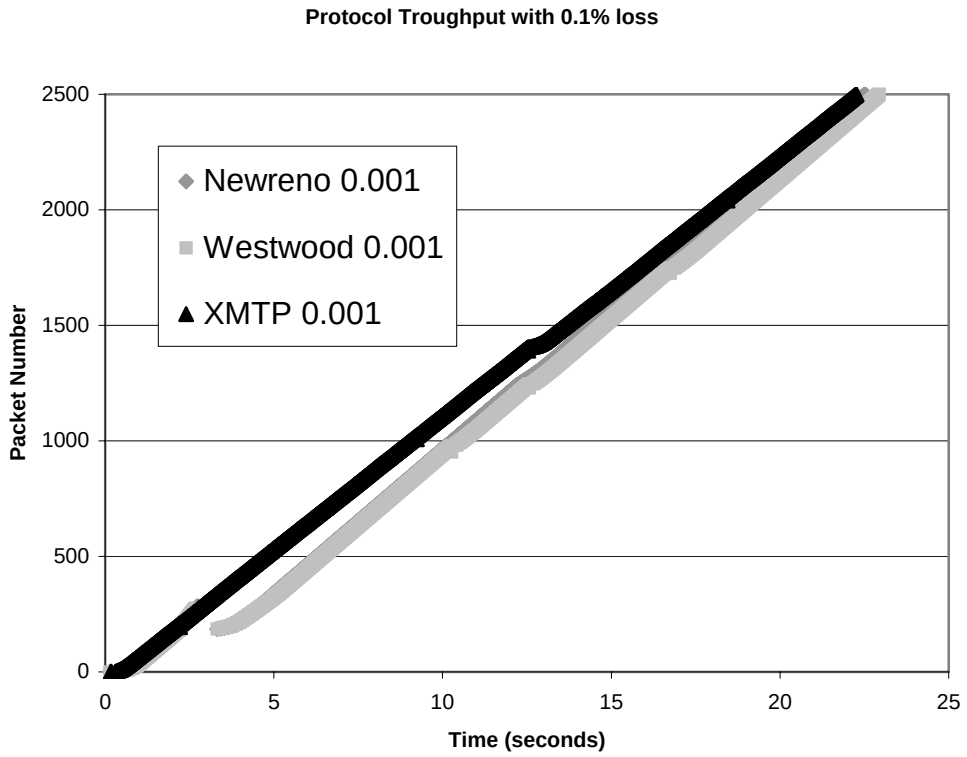



**Figure 27: Protocol Throughput 0.1% loss**

The last simulation in this series shows the results for the three protocols when loss is at 1%. XMTP still performs better than the other protocols. The lost packets changed TCP's behaviour: both TCP flows no longer show the initial congestion loss caused by the slow start algorithm. In the first five seconds, TCP Newreno performs better then TCP Westwood, but in the long term Westwood's loss discrimination allows it to get more packets through.

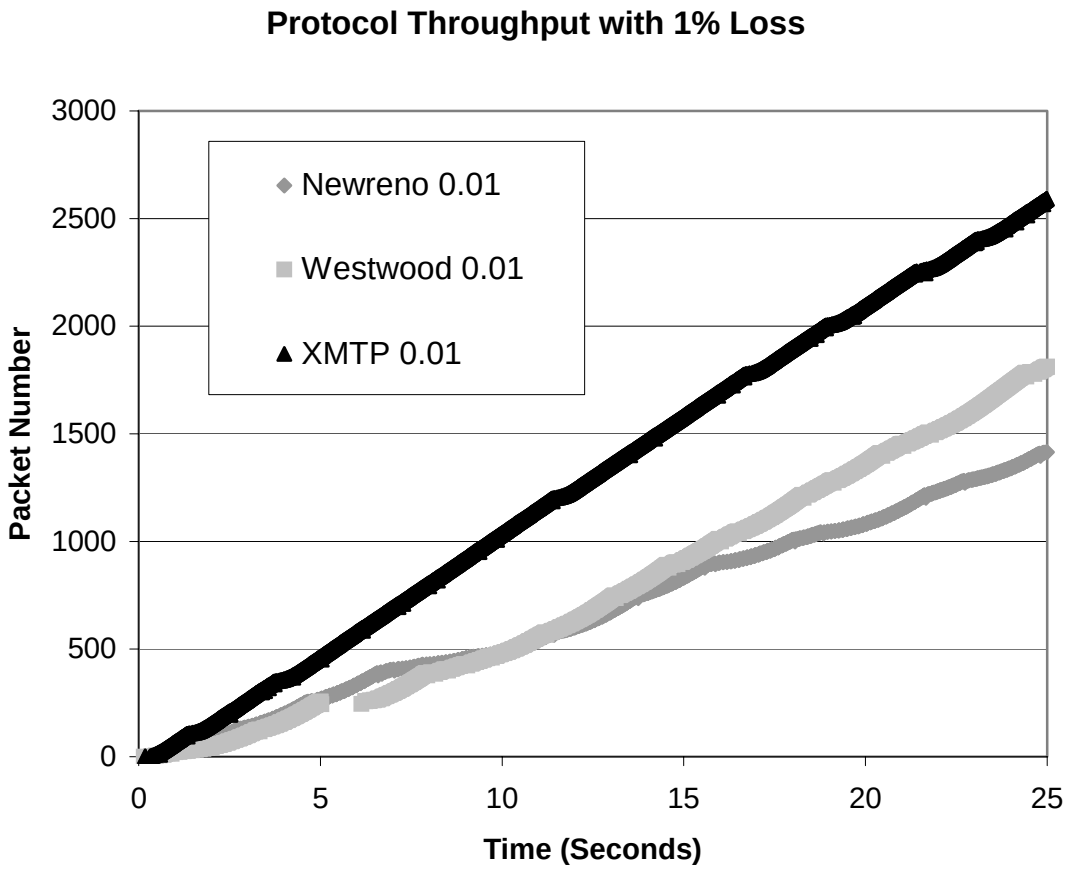
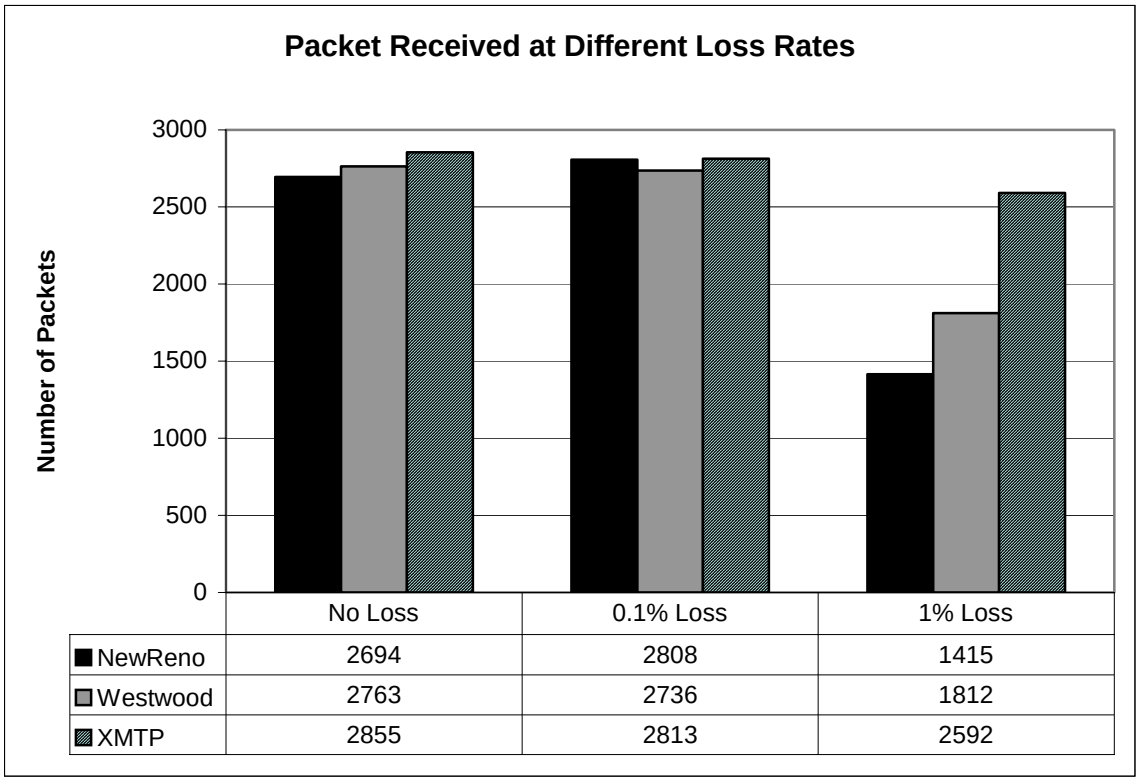



**Figure 28: Protocol throughput 1% loss**

The results are summarized in Table 7. It is interesting to notice that at 0.1% loss NewReno gets slightly more packets though, even though it has no loss discrimination. It is only recovering better from what it believes are congestion losses. At 1% loss, the loss discrimination and fast recovery of XMTP let it get a much better result than both flavors of TCP tested. Westwood gets almost 30% more packets than NewReno, but XMTP is

able to send nearly twice the number of packets NewReno sends.



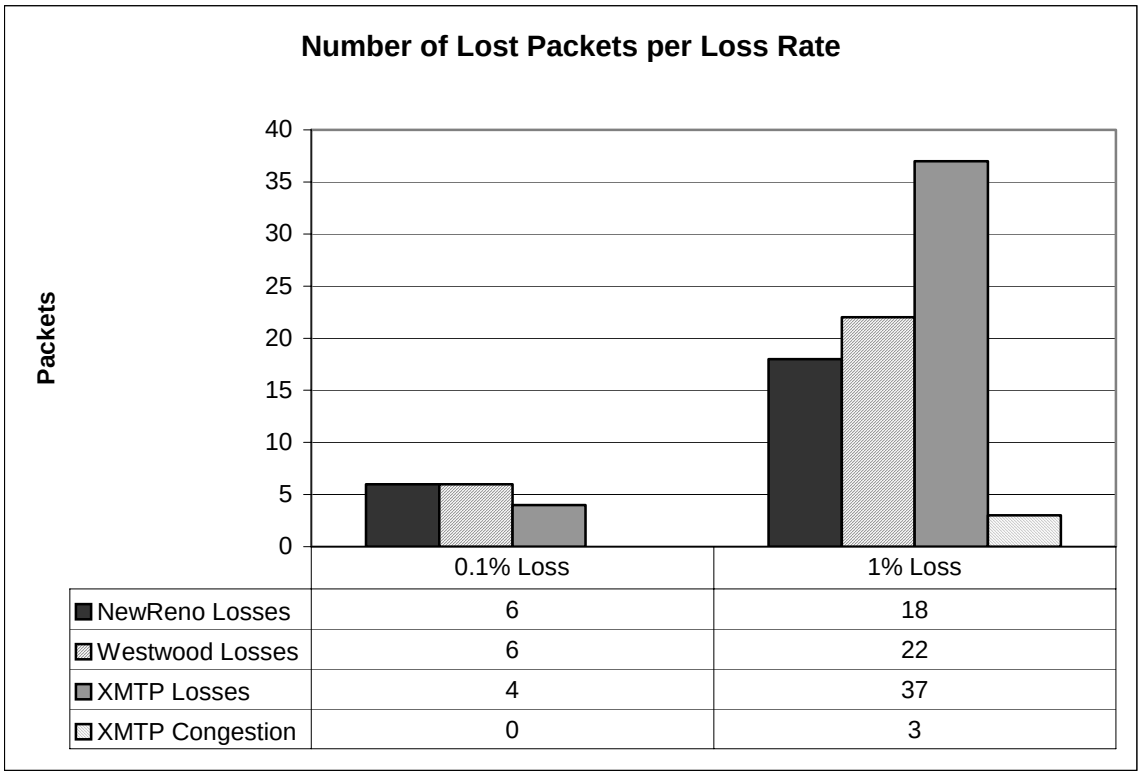

| | No Loss | 0.1% Loss | 1% Loss |
|---|---|---|---|
| NewReno | 2694 | 2808 | 1415 |
| Westwood | 2763 | 2736 | 1812 |
| XMTP | 2855 | 2813 | 2592 |

**Table 7: Number of packets received**

It must be pointed out that the heuristic for preventing flow synchronization implemented in XMTP does not allow it to get the full link bandwidth. Every time XMTP measures available bandwidth to set its sending rate, it adjusts the rate by a random factor, which can be anything up to 10% below the available bandwidth. Statistically, XMTP should be 5% below maximum bandwidth, and it shows on the slope. Although XMTP starts faster because it does not have the initial loss TCP has, TCP grows faster afterwards.

We also measured the number of losses experimented by each protocol during the run

shown in this set of simulations. Those are listed in Table 8. All losses listed were caused by transmission errors. It is interesting to notice that of the 37 losses XMTP had (it had more losses because it sent more packets), only three were miscategorized as congestion losses. All others were recognized as transmission losses, which explains the results that XMTP had compared with both TCP flavors.




| | 0.1% Loss | 1% Loss |
|---|---|---|
| NewReno Losses | 6 | 18 |
| Westwood Losses | 6 | 22 |
| XMTP Losses | 4 | 37 |
| XMTP Congestion | 0 | 3 |

**Table 8: Number of packets lost**

In the last set of simulations, the three protocols are run simultaneously through a 1 Mbps bottleneck, and the packet loss rate is varied. In Figure 29 the results of the run with 1% packet error can be seen.

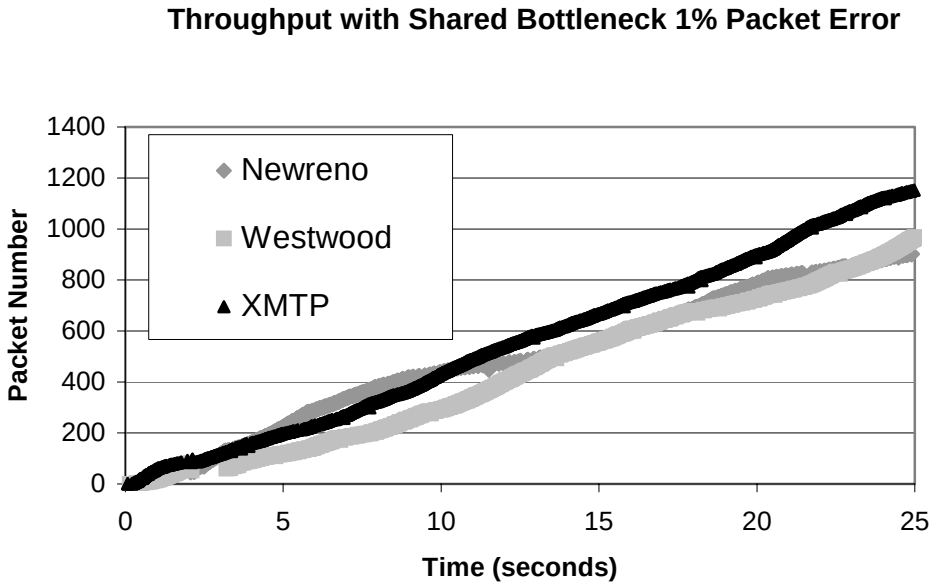
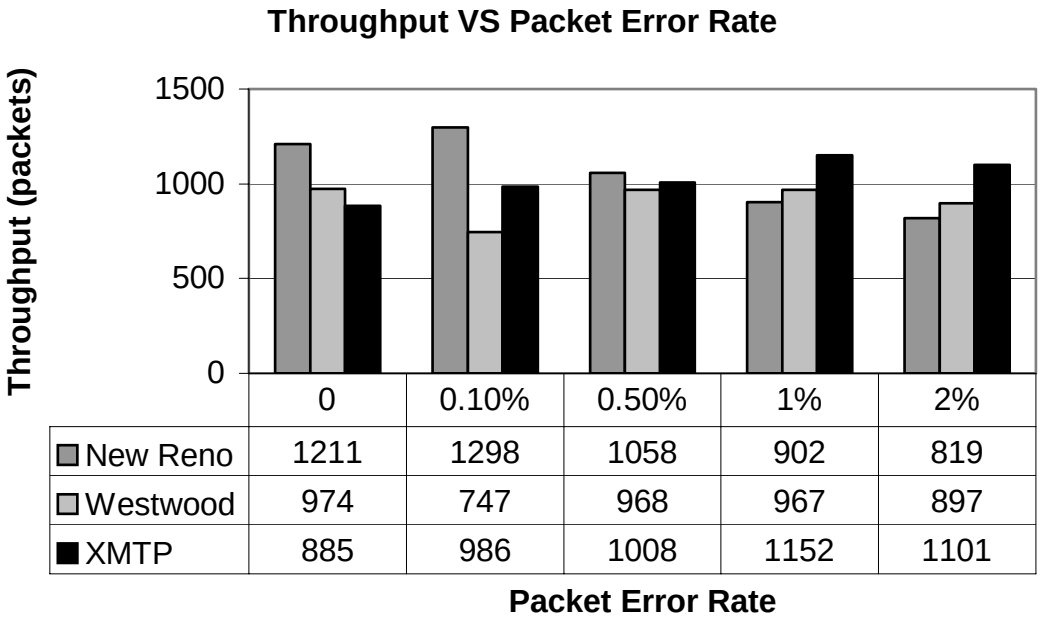



**Figure 29: Protocol Troughput with shared bottleneck and 1% packet error**

Although both XMTP and TCP Westwood have better results than NewReno, NewReno is more aggressive in getting bandwidth than the others, and when there are no packet losses to slow it down it quickly regains bandwidth. XMTP's more efficient loss discrimination allows it to do well at the end, although it is less aggressive than NewReno, and the same can be said for Westwood.

Figure 30 summarizes the results. For low bit error rates, TCP Newreno gets more throughput from the shared link. As the number or error increases, XMTP starts to perform better. TCP Westwood starts to perform better than Newreno at around 1% packet loss, but is not as efficient as XMTP to distinguish between transmission and congestion losses, and does not manage to get as much bandwidth as XMTP.

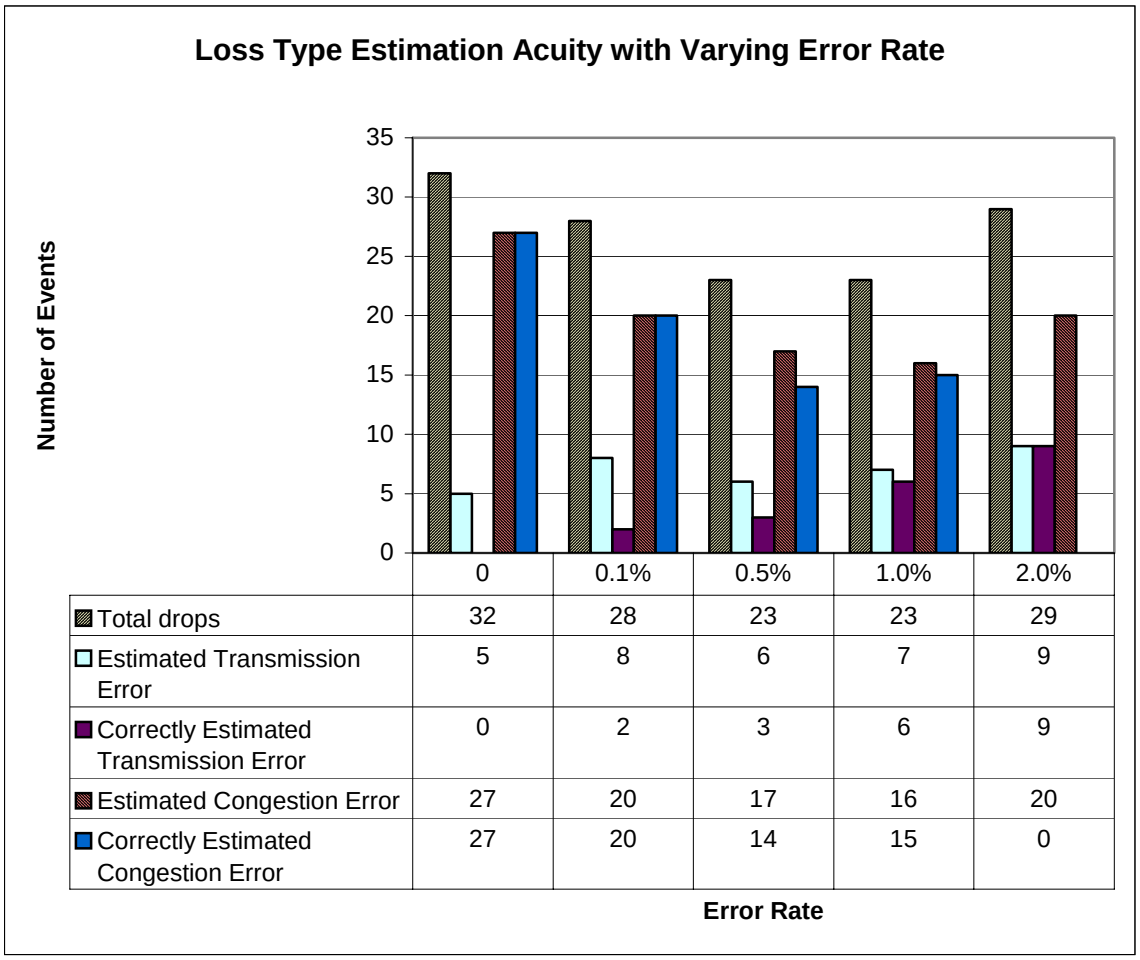



| | 0 | 0.10% | 0.50% | 1% | 2% |
|---|---|---|---|---|---|
| New Reno | 1211 | 1298 | 1058 | 902 | 819 |
| Westwood | 974 | 747 | 968 | 967 | 897 |
| XMTP | 885 | 986 | 1008 | 1152 | 1101 |

**Figure 30: Throughput vs packet error rate**

The results of the loss discrimination heuristic are shown in Table 9. For each packet error rate, the number of lost packets is given, along with how many were transmission errors, and how many were interpreted right or wrong by the heuristic. The same is done for congestion errors. Therefore, when the error rate was 0, there were no transmission errors, but the heuristic guessed that from the 32 errors, 5 were transmission errors. By the same token, the heuristic said that 27 were congestion errors, and all of those were

congestion errors. The heuristic does not have to be perfect for it to work. The homeostatic congestion controller maintains stability when a congestion error is mistakenly understood as a transmission error, and the opposite only diminishes throughput.




| | 0 | 0.1% | 0.5% | 1.0% | 2.0% |
|---|---|---|---|---|---|
| Total drops | 32 | 28 | 23 | 23 | 29 |
| Estimated Transmission Error | 5 | 8 | 6 | 7 | 9 |
| Correctly Estimated Transmission Error | 0 | 2 | 3 | 6 | 9 |
| Estimated Congestion Error | 27 | 20 | 17 | 16 | 20 |
| Correctly Estimated Congestion Error | 27 | 20 | 14 | 15 | 0 |

**Table 9: Summary of loss discrimination heuristic results**

| Packet Error Rate | Bit Error Rate ($\times 10^{-6}$) |
|---|---|
| 0.1% | 0.1125 |
| 0.5% | 0.5625 |
| 1.0% | 1.1250 |
| 2.0% | 2.2500 |

**Table 10: Packet Error Rate to Bit Error Rate Comparison**

The error rates presented in this section are packet error rates. Because one bit of error is all it takes to make the whole packet wrong, the packet error rates tend to be much larger than the error bit rates. Table 10 shows the conversion from packet error to bit error rate.

## *4.6 Conclusions and Future Research*

This chapter presented a loss-discrimination heuristic for rate-based transport protocols. This heuristic can help to improve the throughput of transport protocols in wireless environments, by allowing the transport protocols not to turn on their congestion control unnecessarily. The heuristic was first analyzed by inferring its accuracy on a trace, and then applied to the congestion control algorithm described in the previous Chapter. The modified XMTP was then compared with TCP Westwood, which is also designed for wireless environments. TCP NewReno was used to furnish the baseline.

The main contribution of this Chapter is the description and analysis of the loss discrimination heuristic. It was shown that for lossy environments, XMTP can

outperform both TCP NewReno and TCP Westwood, but for wired environments TCP NewReno has higher throughput (which was expected due to the limit imposed on XMTP to prevent synchronized losses, explained on the previous Chapter).

Chapter 6 contains the evaluation of the loss discrimination using a user space Linux implementation of a reliable protocol using the same mechanisms used in XMTP, but further simulation with larger scenarios is still interesting to investigate if pathological cases can be found where the heuristic will hamper the performance or cause congestion. Due to the requirements of a predictable arrival time, this heuristic seems to be tied to rate-based protocols. It would also be interesting to investigate if it could be adapted to non-rate based protocols by using time-stamps, or even to paced TCP, which would broaden its applicability.

# Chapter 5 MMTP

## *5.1 Introduction*

This Chapter presents the design and implementation of a transport protocol for multimedia traffic with support for mobility using the congestion control and loss discrimination mechanisms described in the previous Chapters. The Multimedia Multiplexing Transport Protocol (MMTP) [MA001] is a rate-based multiplexing protocol designed to carry packets with hard deadlines, to mobile systems that have access to multiple link-layer technologies. MMTP is designed as a multiple channel, rate-based protocol that supports the transmission of time sensitive rate-based data streams that may be generated live or from stored data.

Given the characteristics of the data streams in terms of frame rate and bandwidth requirements, MMTP creates a virtual channel that distributes the multimedia data into any available communication sub-channel. Each sub-channel corresponds to a different interface in the mobile host that is active[16]. As the available sub-channels change, MMTP adapts, adding or removing sub-channels as necessary. MMTP provides a best effort service. If the aggregation of available sub-channels does not provide enough bandwidth for the application stream, MMTP will drop packets that it estimates cannot arrive on time and inform the application of the lack of necessary resources. The main task of

[16] An active interface is one that has connectivity, in the form of a base-station or similar network attachment point, and through which the mobile was able to acquire an unique IP address.

MMTP is the decision as to which sub-channel to use for transmitting the current packet. This decision is based on estimations of the bandwidth and delay characteristics of each sub-channel. After startup, two control mechanisms are used to adapt the sending rate to the sub-channel bandwidth: rate decrease messages and channel probe. Rate decrease messages are sent to prevent congestion when the receiver notices that the channel bandwidth is below the sender's rate. Probing is used to track increases in bandwidth.

The main contribution described in this Chapter is the creation of a complete multiplexing protocol for the transmission of multimedia data in a wireless, mobile environment. The protocol includes multiplexing for higher throughput, retransmissions for partial reliability, a priority model that maps well to transmission of layered data such as MPEG flows and a congestion control scheme that is TCP-friendly. It can be easily used as a basis for a video-streaming application [TE005].

The remainder of this Chapter is organized into four Sections. Section 5.2 presents the design challenges and related research for the design of a multimedia protocol in the Internet as well as in a mobile environment. MMTP is described in Section 5.3, where a formal model of MMTP is presented together with the mechanisms for bandwidth estimation, reliability, flow control and channel management. Experimental results are discussed in Section 5.4. The Chapter ends with conclusions and future research in Section 5.5.

## 5.2 Background and Related Research

The current Internet infrastructure was not designed with the needs of multimedia traffic in mind. The pervading best effort delivery protocol that forms the base of all Internet traffic, IP, has no built in mechanism for reservation of bandwidth or for periodic traffic. The protocols that were developed later to allow the use of the IP infrastructure for multimedia traffic do not consider mobility. Adapting the solutions used on wired hosts to mobile systems is not straightforward, because the characteristics of wireless communication channels are even less agreeable to multimedia traffic. In addition, normal reservation schemes (e.g RSVP [BR097]) used for wired hosts will not work or may become very expensive due to changes in the location of a mobile host. However, varying the quality of the source stream to match the available bandwidth and loss rate has been successfully adapted to the mobile environment, although it falls to the application to keep track of the available bandwidth and other parameters necessary for the adaptation.

Due the periodic nature of multimedia traffic, it is commonly accepted that the best protocols for such data streams use rate-based mechanisms, although even TPC may be used if TCP throughput is more than twice the media bitrate [WA004]. Unfortunately, in wireless links throughput is at premium. Moreover, the lossy nature of wireless communication channels may lower TCP throughput. In response, bandwidth estimation in conjunction with congestion avoidance has been suggested for use with wireless rate-based protocols. One of the earliest examples of a reliable rate-based protocol is

NETBLT [CL088], which was designed for the transport of bulk data and is not suitable for multimedia traffic. Recent examples of other reliable rate-based protocols are WTCP and RAP. WTCP [SI099] is a reliable split connection protocol that has good performance over lossy low bandwidth links that have high latency. RAP [RE099] is a TCP-friendly rate-based protocol for real-time streams, which is very similar in application as MMTP, but it does not have adaptations for mobility or for loss discrimination. The Stream Control Protocol (SCP [ST000]) can also be used in the same scenarios as MMTP, and can be multi-homed. However, SCP does not use multi-homing for bandwidth aggregation, only as a fall-back in case of a failure on the primary link. It can also survive a change of IP addresses, for the same reasons as MMTP (when a protocol is multi-homed, mobility is already built-in).

The aggregation of the bandwidth from two modems has been implemented in both Linux and Windows. In both operating systems, the characteristics of both channels must be the same and only a simple load-balancing algorithm is used for scheduling transmission. The aggregation of many lower bandwidth channels in a larger pipe is called “reverse multiplexing” in ATM [CH098], and is now part of the ATM specification, as it allows a multiplicity of rates and flexibility in allocating bandwidth for commercial services. Some work has also been done in the aggregation of bandwidth in wireless links ([SN099]) by using the facilities of PPP (multi-link [SK096]). The mechanisms in MMTP are more general, working with heterogeneous interfaces. In [ZH004] sharing of WAN connections (cellular) is made possible by cooperating hosts using IEEE 802.11 interfaces, which allow a form of bandwidth aggregation. It is unclear

if in a real environment the result would be as good as in the simulation, because there is no real diversity in the WAN connections, other than spatial diversity. All cellular connections share the same frequency, and would probably be competing for the same bandwidth. [AP004] presents the advantages of path diversity for media streaming. [GO002] shows that better throughput of multimedia data is possible in a ad-hoc network by using route diversity. [GO002] analyses the impact of path diversity for multimedia streams. The use of multiple interfaces for multimedia streaming, such as proposed in MMTP is also studied in [RI005], which shows that path diversity outperforms interleaving in preventing multiple consecutive losses in the media stream.

By uncoupling the transport protocol from the network protocol, transitions from one network to another are very natural in MMTP, and require no switching. The Barwan project presented the concept of “vertical handoffs” [ST098], transitions from one link layer to another. WTCP uses a similar model, when the mobile transitions from one area of coverage to another, there is a handoff and the older connection is relinquished. In MMTP, if an area is connectivity rich and multiple ways to access the infrastructure are available, the activation of a new interface does not cause another to be dropped. The new interface is added to the existing pool.

Adapting the bandwidth requirements of a multimedia stream to the available bandwidth of the channel has been proposed in [BO098]. Because this requires a close interaction between the transport protocol and the coding application, there are many proposals for integrating source coding and the transport protocol. [LE000] proposes transcoding the

source into a non-prioritized packet stream to ensure graceful degradation in the presence of packet loss, and describes a TCP-friendly rate based protocol and the framework for the interaction of the protocol and transcoder. [SE099] proposes modifications to the TPC protocol, a resilient encoder and a rate control algorithm for the same objective. While we do not delve into source coding, MMTP exposes rate changes to the application, enabling adaptation.

MMTP can be viewed as two one-way protocols, one from sender to receiver carrying data, and another from the receiver to the sender, carrying control information. RTP [SC096] uses different streams for data and control information, while MMTP carries control information inside the data packets, allowing for changes in the rate for presentation be communicated to the application simultaneously with the receipt of data. RTP does not implement reliability or congestion control, leaving for the application the task of deciding which level of reliability it wants or needs, and implementing that level of reliability. Direct comparison of RTP and a single channel MMTP should return the same results if the same state machine encoded in MMTP is added to the application. MMTP has the added ability of using multiple channels, which is not supported by RTP.

Because MMTP does not necessarily back-off on lost packets, mechanisms such as RED [FL093] may not affect the sending rate in the same way a TCP stream would be affected. Although we believe that our congestion control method results in fair resource utilization, the work presented on [JE099] on the differentiating multimedia traffic on router to avoid congestion can be used to police MMTP traffic.

A new approach to mobility is to make mobility visible to the endpoint. While Mobile IP tries to hide host mobility by using proxies, mobility can also be achieved by letting transport protocols take care of the switch between access points. This requires the uncoupling of the network address from the connection identifier, and a scheme for location. TCP and DNS were modified to accomplish that in [SN000]. The same end-to-end arguments apply to MMTP, and MMTP has the additional advantage in the case of multimedia traffic because it generally does not need nor works well with the added overhead of TCP reliability.

### 5.2.1 Multimedia in Mobile Environments

Multimedia traffic is very sensitive to delay, jitter and bandwidth restrictions. Introducing wireless links into the path of a multimedia data stream not only increases the potential for such problems, but also brings with it problems from handoffs. In an effort to offset these negative effects, a mobile host may have access to multiple communication technologies. With the growing pervasiveness of the wireless infrastructure, in many places there will be an overlap of coverage. This presents an opportunity to tap additional resources to help the transmission of multimedia traffic. At the same time, mobility-aware protocols may make up for communication artifacts created by movement, such as variations in bandwidth and changes in channel availability.

Consider the following scenario: a user is waiting for a train and wants to watch the

news. The train station terminal has both short-range radio and infrared connectivity. The user has a cellular modem in addition to the infrared and short-range ratio interfaces on the palmtop. Currently, the user would have to choose which interface to use to access his/her favorite news source. Even if the user chose the interface with the best qualities, any fluctuations in service characteristics would directly impact reception. When the user boards the train, the local connectivity (short-range radio and infrared) becomes unavailable. If the user had chosen any of the local connections, there would be a gap in the transmission as the connection was being handed off to the cellular modem. The solutions presented in this work improve this scenario in two ways: better channel quality and seamless handoffs, by using connection diversity [TO003].

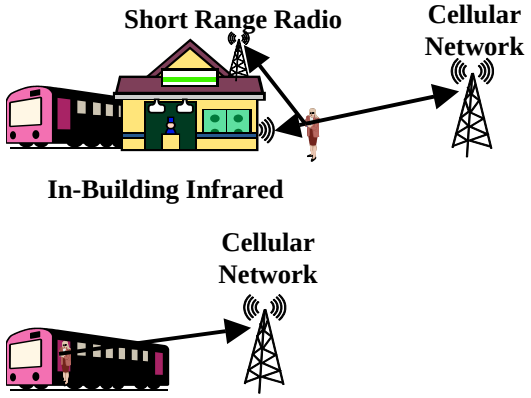



**Figure 31. Wireless Mobility**

The improvement of quality comes with the use of all available channels. This not only aggregates bandwidth, permitting a better quality in the multimedia data stream, but also smoothes out variations in a single channel that would impact data presentation by spreading consecutive packets in different media. Exposing the underlying link layer to

the transport protocol enables the simultaneous use of multiple channels. By creating a transport protocol that is mobility-aware, and knows the existence of multiple interfaces, it can offer the illusion of a single “fatter” pipe with better transmission qualities to the multimedia application. This aggregated channel has interesting qualities. Although the individual channel with longest propagation delay dominates the propagation delay for data, as will be shown in the next section, the control data may take advantage of the channel with lowest propagation delay, helping adaptation.

The use of overlapping communication channels allows for smooth handoffs as the mobile node migrates between coverage areas. If the user moves slowly out from an area covered by a certain communication means, the degradation in quality is perceived by the protocol, and the channel is slowly phased out, with no interruptions. When entering an area covered by a new link layer, the protocol adds the new channel to its processing, with the entailing gains in quality.

## *5.3 Modeling MMTP*

MMTP is a real time protocol for multimedia content, aware of frame deadlines and of differing priorities that are assigned to different frames. Because it is wasteful of network resources to transmit a frame that will arrive past its deadline, frames that cannot arrive on time for display will not be transmitted. In the same token, frames tagged with higher priorities are transmitted ahead of lower priority frames. The existence of lower priority frames enqueued means that not enough bandwidth is available for transmission at full

source rate, and lower priority frames will be dropped ahead of higher priority frames. The application has the task of setting frame priorities, based on its content, informing MMTP when the frame is given to MMTP to be sent. For instance, a MPEG I-frame should get a higher priority than a P-frame.

MMTP provides the mechanism where the application can select how each frame it is sending should be treated. The application should tag its traffic in such a way to get the results it needs, that is, to choose the policy it will use. It may have a MJPEG source, where each frame is of equal importance, and so all frames may be tagged with the same priority. On the other hand, it may choose to alternate between two priorities (0 and 1, for example), so in the event MMTP has to drop frames for lack of bandwidth, the probability of two consecutive frames being dropped is lowered. The assumption is that the application knows the content, and so has better knowledge of what should be dropped in the case the need arises.

To be able to send frames according to the bandwidth availability in each sub-channel, MMTP has to measure the bandwidth along each path associated with it. This is done through the implementation of passive measurements and a probing mechanism to assess the bandwidth and traffic load of each link. Each sub-channel measures its bandwidth independently. Frames are sent periodically, and the receiver knows both the period in which each frame was sent and their arrival times. This way the lengthening of the period can signalize diminishing bandwidth. To measure increases in bandwidth, from time to time each sub-channel will send two consecutive frames back-to-back and measure their

inter-arrival time on the reception, adjusting the parameters on each end if needed.

A model of the MMTP requires both the description of its interface and classes and diagrams giving emphasis to certain aspects of the protocol's real time nature. The Unified Modeling Language (UML) [OM001] offers two types of diagrams that are very useful in that regard. The first is the classes model, with the different classes present in the protocol shown with their most important attributes and methods. The classes model also shows the relationships between objects of these classes. This type of model can be used to guide the implementation of the protocol within different programming environments.

In Figure 32 we have the class model for MMTP. The application opens a connection through the MMTP API, and sends and receives frames. The System detects the availability of active interfaces and adds or deletes interfaces as it acquires or loses IP addresses. Each SubChannel is associated to a different interface through a unique IP address, and MMTP frames are encapsulated in IP packets.

The second type of diagram, the state diagram, is useful to describe the states of the classes and the events that will evoke methods during run time, as well as show the existence of more than one thread running simultaneously. The states are chosen to best represent some aspects of the internal mechanisms of the protocol, such as probing and the queue interface. A possible application of the state diagrams description is a direct translation of its states to a formal state machine description language such as ASM

(BO099], [GU000]), in which form it can be tested using formal verification tools.

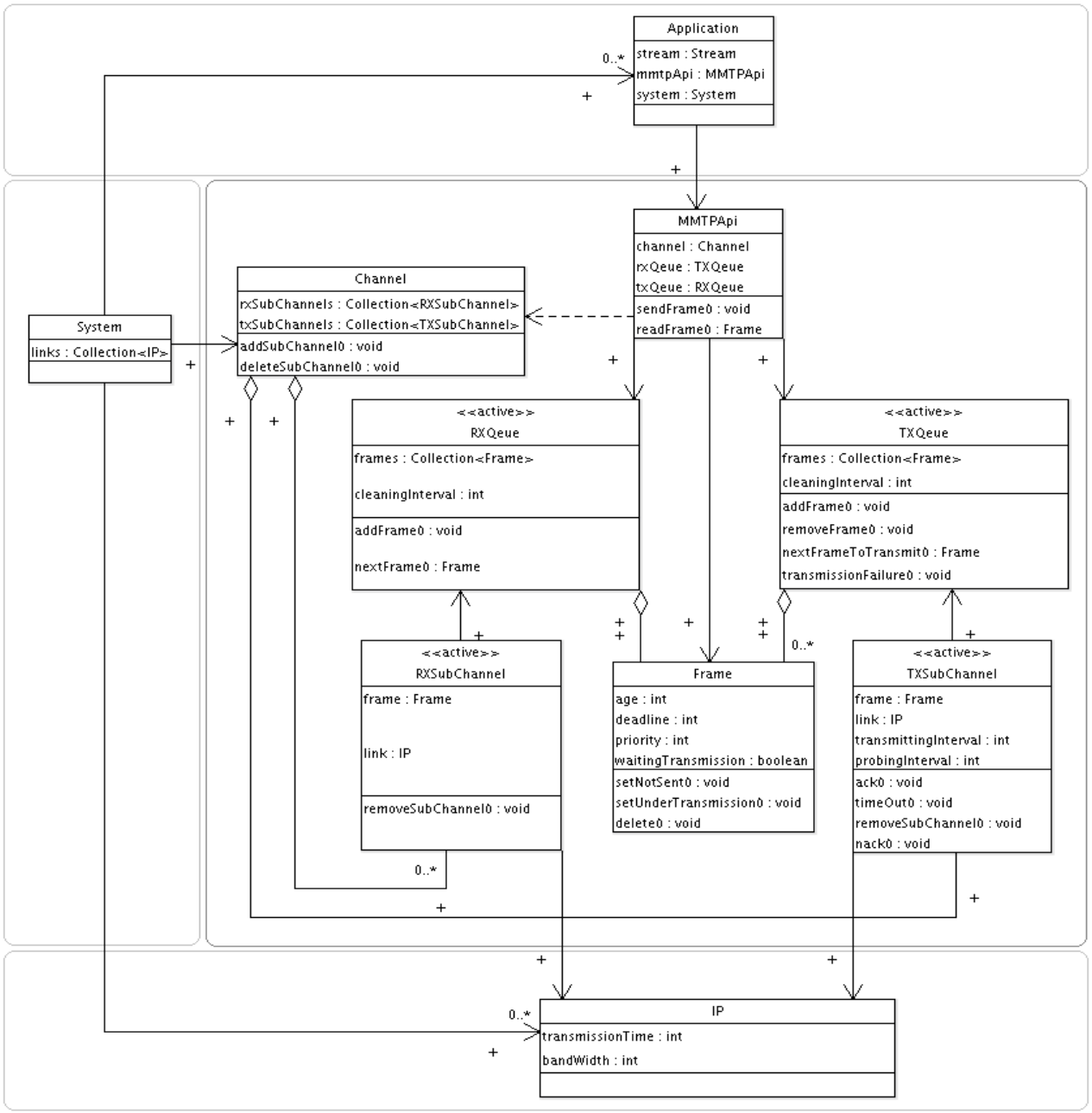



**Figure 32: MMTP Class Model**

The first class in the model is the MMTPApi, representing the interface to the application, shown on Figure 33. It will start a Channel instance and two queues, one for transmission and other for reception. After this it will stay on its first state, Ready, waiting for requests for sending or reading frames, to which it will respond accordingly by changing its state and querying the appropriate queue.

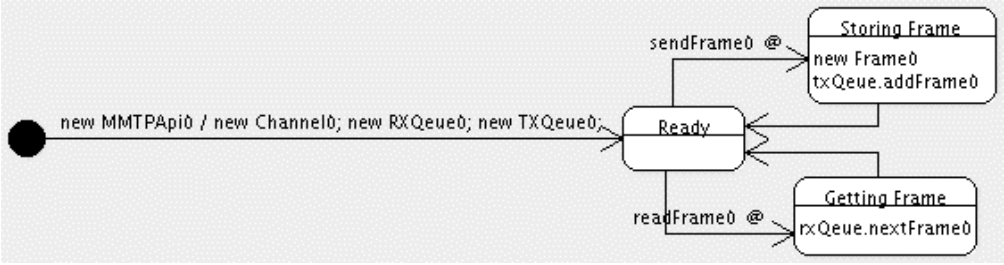



**Figure 33: MMTP API**

The next class is the Channel (Figure 34), which is also not an active class. It will respond to external stimuli from the system's LLM (discovery or removal of links). Every time a new link is acquired or lost the Channel will change its state from Waiting Changes to Adding or Removing SubChannels, making or destroying an instance of RXSubChannel and another of TXSubChannel.

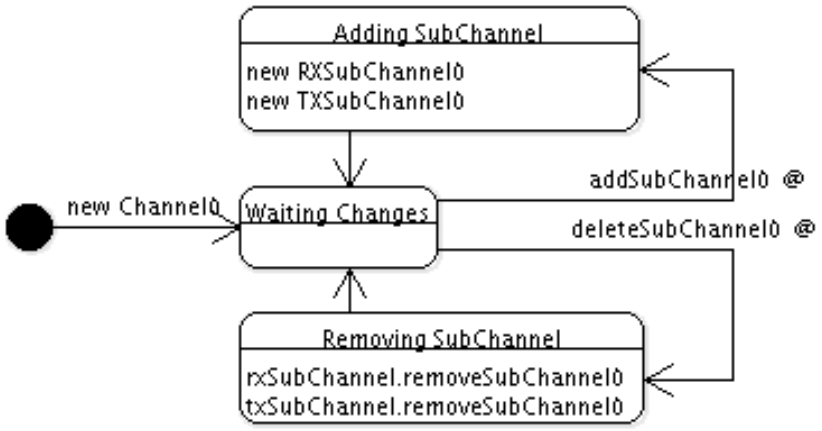



**Figure 34: Channel**

The SubChannel classes are responsible for delivering frames stored on the queue to be sent, as well as handling probes. The TXSubChannel has MMTP's most complex state diagram (shown on Figure 35). It has two threads; one responsible for periodically transmitting frames and probes, another for listening for acknowledgments, timeouts and parameter changes from probing measurements. When a Frame is properly transmitted, it is removed from the queue. But when transmission fails for any reason, including the loss of the link, the queue is informed to place the Frame back on queue for transmission. A similar role is performed by the RXSubChannel (shown on Figure 36), which will listen for arriving frames and probes, storing the first and using the second to calculate the link new parameters (for instance, a new transmission rate). These parameters are communicated to the sender through the acknowledgments sent back from the receiver.

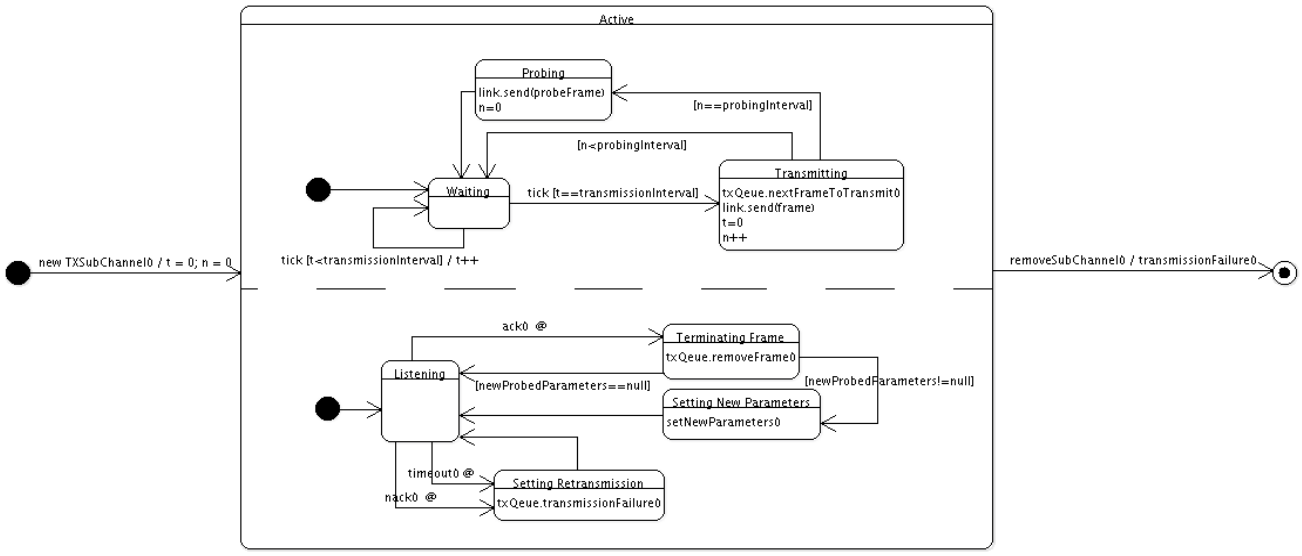



**Figure 35: Transmission Channel TXChannel**

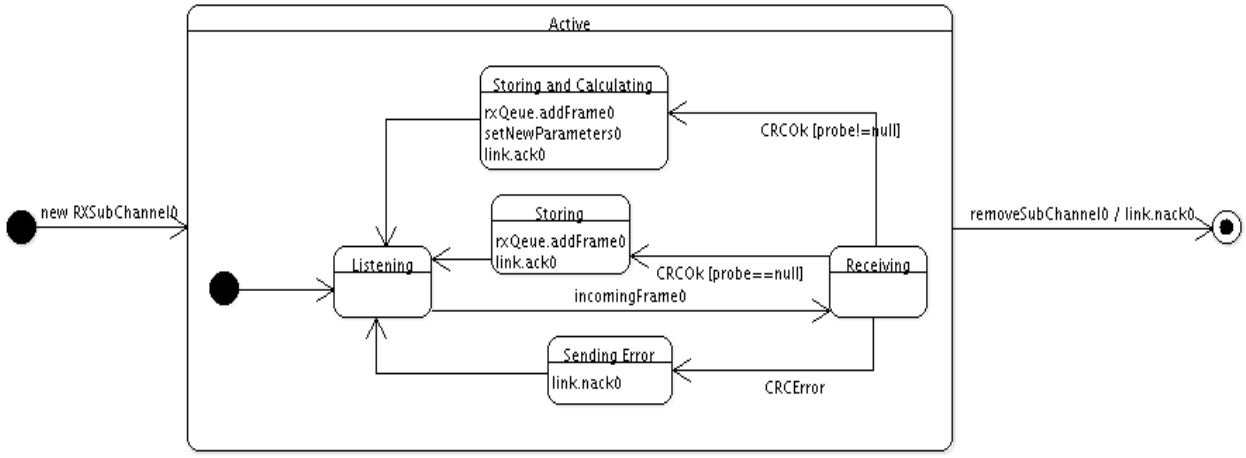



**Figure 36: Receiving Channel RXChannel**

The Frame (Figure 37) is simply a container for multimedia content, along with priority

and deadline parameters. The Frame has two possible states, Waiting or Under Transmission, and it can be terminated by the queue at any given time after its deadline.

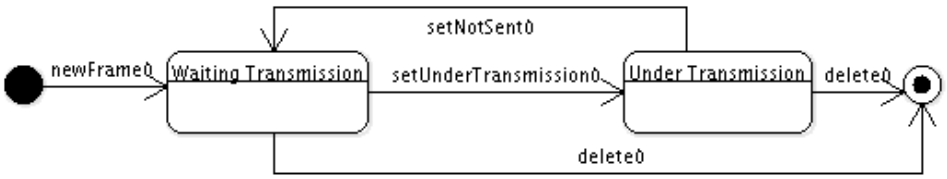



**Figure 37: Frame**

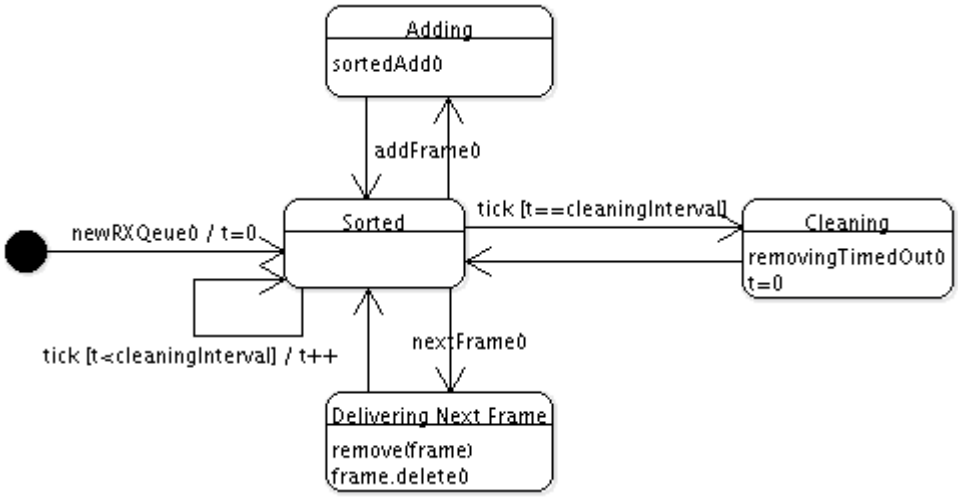



**Figure 38: Receive Queue RXQueue**

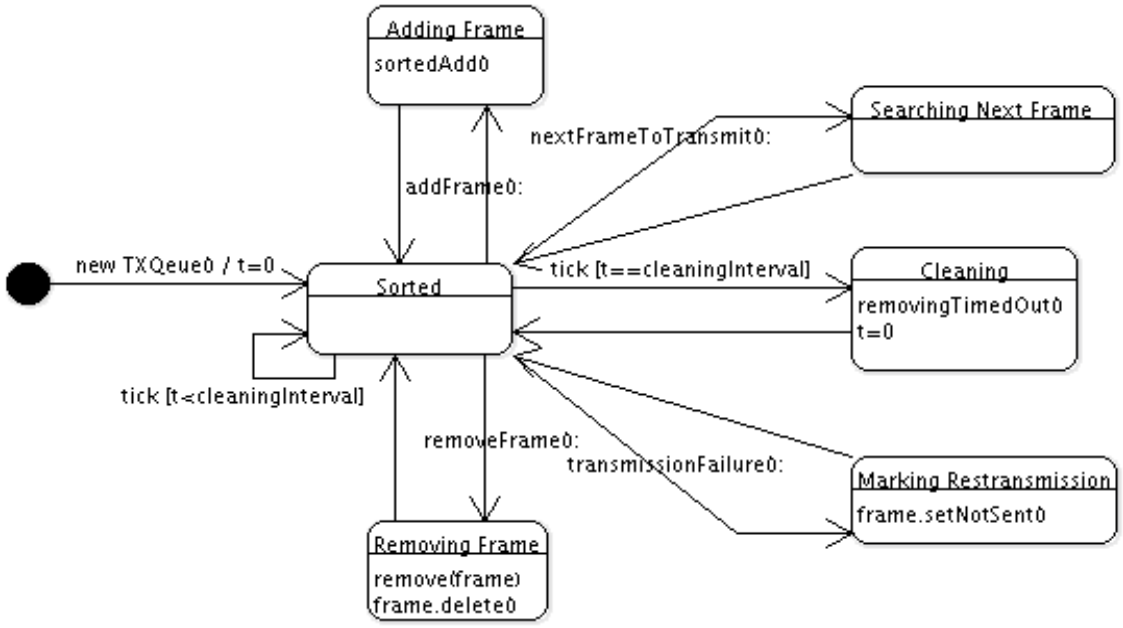



**Figure 39: Transmission Queue TXQueue**

Finally, the queues (Figure 38 and Figure 39) are modeled here as shared protected memory queue managers, very similar to processor access queues. They execute SubChannel and MMTPApi method calls in a synchronous manner, and the implementation should be able to handle concurrent calls properly. Each method will change the queue state and perform the corresponding action. For instance, the method transmissionFailure() of the TXQueue will inform the Frame to switch back to its Waiting Transmission state. The decisions based on frame priorities and deadlines are implemented in the sortedAdd() and nextFrameToTransmit() methods, as well in the queues active periodic cleaning thread.

### 5.3.1 Data Characteristics

Multimedia data streams can be generated on the fly at the rate at which they need to be played out or retrieved for storage. Applications using on-the-fly streams often have very little tolerance for delay, while applications using stored media may be more tolerant. In addition, for the former, frames are only accessible for transmission as they are generated, while in the later, all frame are accessible at the same time. Frames from multimedia data streams often exceed the maximum transmission size and must be fragmented into multiple packets. Intelligent fragmentation can be done that enables the reception and processing of pieces of the frame, even if the entire frame does not arrive, supporting the concept of application-level framing [CL090]. In the rest of this Chapter, we discuss MMTP in the context of on-the-fly data with one packet per frame.

### 5.3.2 Startup Behavior

On startup, the application defines the requested *frame rate*. The protocol queries the Link Layer Manager to learn the number of available channels, and finds an estimate of the *propagation delay* and *packet rate* for each. To see if the required *frame rate* can be met, the protocol calculates the available aggregated channel rate. There are two cases:

1. ***Frame rate*** **> $\sum$ packet rate(i)**
    - the application is notified that packets will be dropped. The application may decide to abort the transmission, to change the *frame rate* or just continue.

2. ***Frame rate*** **< ∑ packet rate(i)**

    - the estimation indicate that there is enough bandwidth for the transmission.

*Initial delay* is a playout parameter that tells the receiving application how long to wait before playing the first frame. *Initial delay* can be chosen between the *maximum initial delay* given by the application and the *minimum initial delay* calculated by the protocol given the *propagation delays* of active channels. The larger the *initial delay*, the more slack the protocol has to compensate for jitter, because it adds time to the deadline of each frame. The receiving application will buffer a number of frames proportional to the ratio between *initial delay* and *frame period* (the inverse of *frame delay*) before initiating playout.

*Initial delay* is always greater or equal the longest propagation delay for the channels being used and is not dependent on which channel is used to send the first packet. The examples in Figure 1 depict the startup behavior assuming two channels and one packet per frame. Figure 1a shows *initial delay* when the first packet is sent on the channel with longest propagation delay, and Figure 1b shows *initial delay* when the first packet is sent on the other channel. In the both examples, the sender receives frame 0 at time *t* and frame 1 at time *t + 1/frameRate*.

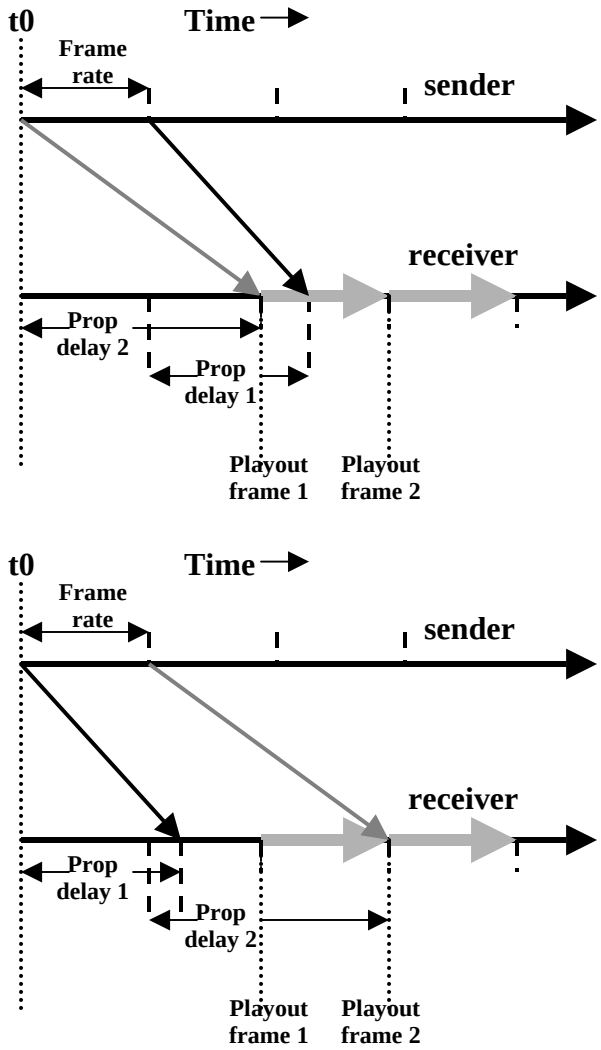



**Figure 40. Startup Behavior for MMTP with Two Channels**

In Figure 1a the first frame was sent on the channel with longest propagation delay, so playout can start as soon as frame 0 is received, since we know that frame 1 will have arrived at the end of frame 0 play period. If propagation delays of the two channels are very different, it is possible that frame 1 will arrive before frame 0. In this situation,

frame 1 is buffered and playout still begins upon receipt of frame 0. In Figure 1b, frame 0 arrives first, but playout must be delayed until within one play period of the receipt of frame 1, or else there may be a gap at the end of the playout of frame 0. Because MMTP uses estimation for both *available bandwidth* and *propagation delay*, it is difficult to implement option 2, since *initial delay* depends directly on the estimated value for *propagation delay* in both channels. Sending the first packet on the channel with longest propagation delay allows the playout to be self-clocked

### 5.3.3 Flow Control

Flow control for MMTP can be modeled as a set of rate control protocols, one for each channel, all servicing a single queue. Packets are inserted into the queue at the frame rate of the source and need to be transmitted on one of the available channels. This type of resource management problem closely models the scheduling of processes in real-time multiprocessor operating systems [JE085], where processing time is mapped to transmission time for each channel. Packets are transmitted on a channel when a time equals the current period has elapsed since the last transmission and if the *propagation delay* ensures the frame will arrive on time.

When a frame arrives, there are three possibilities:

1. No channel is ready: the frame has to wait until a period has elapsed in one channel that will deliver it on time. If no channel is ready before the frame's deadline expires, the frame is discarded.

2. Exactly one channel is ready: if the channel can deliver the frame on time, the frame is sent. Otherwise, this case reduces to case 1.
3. Multiple channels are ready: if more than one channel can deliver the frame, the channel with longest propagation delay is chosen. Maintaining the channel with longest propagation delay filled helps creating a fast response path. If no channels can satisfy the frame's deadline, the frame waits as in case 1.

Even if none of the available channels can currently send a frame, queued frames may still be transmitted if a new channel with smaller propagation delay is added or a large decrease in the expected propagation delay of one of the current channels enables successful packet transmission.

### 5.3.4 Congestion Control

MMTP uses the Homeostatic Congestion Controller, described in Chapter 3, and the loss discrimination heuristic described in Chapter 4. Congestion control, therefore, is implemented as a reactive technique based on bandwidth estimation.

To guarantee the arrival of a frame in time, MMTP has to keep track of the propagation delay in each of its sub-channels. Both *propagation delay* and *available bandwidth* in a sub-channel will vary due to routing changes and due to interference caused by other traffic. The *available bandwidth* is equal to the difference between the maximum bandwidth of each link and the usage of each link. *Propagation delay* is composed of two parts: the actual propagation delay of the bits in the transmission lines and plus the time

spent during processing in each router on the path. The first part is fixed in the absence of route changes, but the second will grow according to the size of the queues in the intervening routers.

To avoid congestion, the protocol tries to keep the requested bandwidth below *available bandwidth* in a channel. This is done by measuring *available bandwidth* and changing the rate packets are sent to a compatible value. *Available bandwidth* is inferred by measuring the inter-arrival times of packets at the receiver, and feeding the measurements back to the sender. Because packets are being sent at regular intervals in each channel, the inter-arrival time should converge to the period frames are being sent on that channel if sufficient bandwidth is available. If the inter-arrival times start to grow, somewhere in the path a router is running above capacity, and queuing packets.

#### *5.3.4.1 Parameter Estimation*

To estimate the basic parameters, *available bandwidth* and *propagation delay*, the inter-arrival time of packets is tracked at the receiver for each channel. Inter-arrival time is compared to the period frames are being sent on that channel (present on each packet header). With stable queueing delays, inter-arrival time should converge to this period. As queueing delays change, jitter is introduced and inter-arrival time will vary. A running total for jitter should be zero if there are no changes in channel characteristics, stabilizing the average inter-arrival time. A packet that arrives late causes the next packet to appear to arrive earlier, so the sum of the jitters should cancel. The accumulated jitter is a moving average of the last n packets, where n is an implementation parameter. The

size of n should be such that it has the same temporal granularity of the keep-alive messages, so those can carry meaningful data back to the sender.

There are two special cases when running total for jitter will not be zero:

- When there is incipient congestion, the queue sizes on the routers grow, and this is seen at the receiver as a positive increase in the accumulated jitter. In this case, the receiver sends back a message asking for a decrease in the sending rate, so the transmission will not cause congestion. This is actually a better mechanism than decreasing bandwidth usage upon dropped packets. Using dropped packets as a measure of congestion is a reactive technique, used when congestion is already present. By using the accumulated jitter to signal approaching congestion, the protocol can try to prevent packets from being dropped. Moreover, dropped packets are not a good measure of congestion for mobile hosts, where wireless links have much higher loss rates than normal wired lines.
- When propagation delay drops, there will be a drop in the inter-arrival time in one time-slot. After that, the perceived rate will again converge to the sending rate. To detect that this drop is not an artifact cause by a previous packet that was very late, the jitter history is analyzed. If the pattern shows a large positive value followed by a large negative value (which would cancel out), then this event is ignored. If the trend is stable, and the early packet is not an artifact, then the next keep alive message will carry an indication that extra bandwidth may be available on that channel.

The measurement of inter-arrival times works well to prevent congestion, but it does not work to measure excess or idle bandwidth in a channel. The problem is that at every sending rate, the maximum bandwidth measured by the application is equal to the bandwidth being pumped at the sender. Therefore, we can decrease the sending rate, but not increase it with this method. Another mechanism is needed to measure available bandwidth above what is being used.

#### *5.3.4.2 Probing*

The mechanism MMTP uses to measure an increase in available bandwidth is probing. There are two easy ways to implement probing. The first one is additive increase: to continually increase the send rate by small amounts in the absence of rate decrease requests, and keep normal inter-arrival time measurements. If the inter-arrival times start to grow, that means the optimal rate was overshot or the network is experiencing congestion, and the protocol has to throttle back the rate. The problem with this approach is that the virtual bandwidth gains are not tested until needed, as the packet rate is bound by the frame rate of the multimedia stream. In addition, when needed the virtual bandwidth measured by the rate increases may not be there. The second is to use probe-packets. A probe packet is sent back to back with a normal packet. The inter-arrival time of the packet and the probe should be zero, as they were sent back-to-back. However, normally this value will not be zero, but instead measure the queuing delay inserted by the routers.

Probe packets should carry useful payload, as they will take place of a normal packet.

The problem is that there may not be data available to send two packets back-to-back. In this case, an empty packet can be sent. This type of probing can be harmful if the system is working at the limit: the bandwidth lost to the probe packet may cause a useful packet to be dropped.

### 5.3.5 Adding and Removing Channels

The initial list of available channels is received from the Link Layer Manager when the protocol starts. As time goes by, new channels may become available, and old channels may be lost. The communication framework notifies MMTP of those events, and gives ancillary information as estimates of *available bandwidth*.

When a new channel is added to protocol processing, the *available bandwidth* has to be measured if this data is not present. To do the initial probing, the protocol uses the packet pair method [KE092], the same mechanism as the normal protocol probing implemented in the Homeostatic Congestion Controller: two messages are sent back to back and their inter-arrival time is measured at the receiver. This is used as the estimation of maximum bandwidth on that link, and fed back to the sender. When a channel is lost, a de-registration message is sent to the peer using the channel with lowest delay. This takes out that interface from the protocol processing. The message also contains the last packet received on that channel. Outstanding packets are put on the queue for retransmission, with the same constraints of the other packets (they will not be sent if they cannot meet the deadline.)

Every active channel sends keep-alive messages periodically. Besides notifying the peer that the channel is still open, these messages carry updates of the measures performed at the receiver, the inter-arrival times of the messages, the estimated *available bandwidth*, the number of packets late and lost. If a keep-alive is not received, the sender sends a query message to assure that the channel is still open or if a control message was lost.

A channel may be present in processing but not carry any data if the channel delay is greater than the slack on the channel. This may happen if transmission started before the channel was added to the processing. If the delay on the new line is greater than the initial delay, the channel can never be used to carry data for that transmission. It can still carry control messages, but it will only be used if no other channels are available. Normal keep-alive messages and probing are done in the channel, in case the delay drops to a usable level. A channel may also be phased out if the delay or bandwidth drops to very low levels. This may happen as the user moves out of the coverage area and the link layer conditions deteriorate, or by congestion. While no de-registration message arrives, keep-alive and probing will continue. If only one channel is available and keep-alive messages are not being received, the protocol will signal that the link may be broken.

## *5.4 Experimental Results*

The experiments show two important aspects of MMTP. The first is bandwidth aggregation: it proves that better quality streams can be sent if more than one channel is used. This was accomplished by measuring the number of packets that arrived on time.

The second aspect of MMTP is the economy of bandwidth. MMTP will only send packets that it estimates will arrive within the packet deadline. This way no resource is wasted. This is shown by the goodput, or ratio of packets that arrived on time by the total number of packets sent. This was shown indirectly by measuring the number of packets that arrived late.

### 5.4.1 Experimental Testbed

The test setup consists of two Sony laptops, a PCG-505TR and a PCG-F450, both running RedHat 6.2 linux with 2.2.15 kernels, patched for infrared, and with PCMCIA package version 3.1.20. The first is connected to the network using a 3Com 574 PCMCIA adapter at 10Mbps. The second has two connections, an IRLan connection to an HP NetBeamIR using an Actisys 2000+ dongle on the serial port, and a Rangelan2 PCMCIA adapter, as shown on Figure 3. The first laptop is running the proxy software and the mobile the client side.

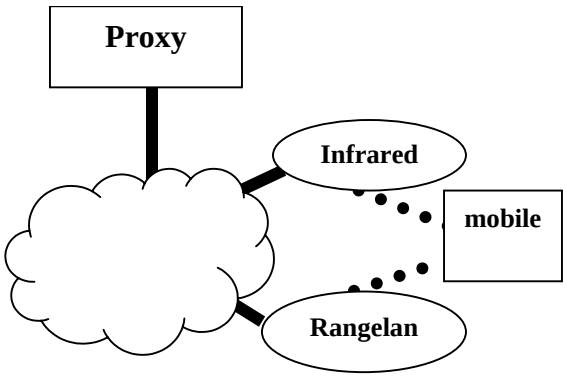



**Figure 41. Test Setup**

Infrared and Rangelan have very different characteristics. Infrared offers a point-to-point reliable link layer, with link speeds up to 4Mb/s. The use of a serial dongle limited the speed to 115Kbps, and the protocol overhead further limited throughput to 9KBps (more than 10% overhead due to the half duplex channel, and reliability and framing added to the infrared link). Rangelan is an old radio technology, the radio link is subject to burst errors, and throughput varies with medium usage. The best throughput measured was on the order of 36KBps. The major characteristics are given in Table 11.

| Delay for packets of 1400 bytes | Minimum (msec) | Average (msec) | Throughput (KB/sec) |
|---|---|---|---|
| Infrared | 315 | 465 | 9.2 |
| Rangelan | 176 | 291 | 36.4 |

**Table 11. Characteristics of the Wireless Networks**

We assume that there is a source generating frames of 1400 bytes with a certain periodicity. We tested our protocol against a program that sends frames at CBR using UDP as they are being generated. The results for the test program follow.

### 5.4.2 Rangelan

These are typical results since there are small fluctuations due to traffic and errors on different runs. For each run, the source program generated 1000 frames of 1400 bytes each. A frame was generated every x microseconds, as shown on the table. Frames were deemed late if they arrived more than one second after they were generated. As an example, if our cutoff value was 6 seconds, all frames on the run with period equals

35000 microseconds would have arrived on time, as this is the largest value for the delay, and no frames were lost. Of course, this is a limited run, and for unbounded media if there is not enough bandwidth the delay would keep on growing, making frames late. Frames are normally lost in bursts – the program recorded both the numbers of bursts (blocks of lost frames) and the total number of frames that were lost.

| Periodicity (in msec) | Late frames (delayed more than one second) | Lost frames/ blocks | Frames that arrived within one second | Total frames that arrived |
|---|---|---|---|---|
| 10000 | 10 | 684/35 | 306 | 316 |
| 15000 | 582 | 352/37 | 66 | 648 |
| 20000 | 6 | 350/38 | 644 | 650 |
| 25000 | 753 | 68/10 | 179 | 932 |
| 30000 | 0 | 101/15 | 899 | 899 |
| 35000 | 818 | 0/0 | 184 | 1000 |
| 40000 | 18 | 20/3 | 962 | 980 |
| 50000 | 1 | 21/3 | 978 | 979 |
| 100000 | 0 | 17/1 | 983 | 983 |

**Table 12: UDP CBR on Rangelan**

It is interesting to note that the number of “good” frames seesaws as the periodicity varies (as seen on Table 12). For a multimedia flow, late frames are useless, so the important number is given on the fourth column – even though more frames may have arrived, if they are late they are wasted. A frame that arrived late also used bandwidth and time – making other frames that follow it late. Ideally, dropping frames should happen at the

source. Another way of using more frames is changing the allowed delay, by buffering frames so the deadline of each frame is postponed. In some runs, the UDP frames flooded the wireless link, and these frames were dropped, allowing more frames to arrive in time.

### 5.4.3 Infrared

Because infrared offers a reliable link, all dropped frames are the result of UDP flooding. The maximum delay on infrared is smaller than on Rangelan, due probably to a smaller buffer on the base-station. If the cutoff were 3.5 seconds, all frames that arrived would be on time. The arrival of the frames was a monotonous increasing function, the first frames arriving on time until the 1-second cutoff was exceeded, and all subsequent frames were late.

| Periodicity in (in msec) | Late frames (delayed more than one second) | Lost frames/ blocks | Frames that arrived within one second | Total frames that arrived |
|---|---|---|---|---|
| 10000 | 84 | 909/64 | 7 | 91 |
| 50000 | 346 | 646/322 | 8 | 354 |
| 100000 | 671 | 312/309 | 17 | 688 |
| 150000 | 309 | 0/0 | 691 | 1000 |
| 160000 | 0 | 0/0 | 1000 | 1000 |

**Table 3. UDP CBR on Infrared**

As soon as there is enough bandwidth to carry the traffic, both losses and late frames drop to zero. The intervals are shown are not the same as Rangelan due to the differences

in bandwidth and delay.

### 5.4.4 MMTP

The test setup for MMTP has a source process that creates frames at the given periodicity and sends them to the proxy. The proxy sends these frames to the receiver on the mobile. The client process does not record the same information on lost frames, so the third columns of the tables are not comparable. Most of the frames shown on the third column were discarded at the source and not lost in the network. Changing the cutoff time would change the number of sent frames, as the protocol would assume that more frames could meet their deadline.

| Periodicity in (in msec) | Late frames (delayed more than one second) | Lost/ discarded frames/ blocks* | Frames that arrived within one second | Total frames that arrived |
|---|---|---|---|---|
| 10000 | 0 | 761/21 | 178 | 178 |
| 15000 | 25 | 562/17 | 413 | 438 |
| 20000 | 0 | 548/27 | 452 | 452 |
| 25000 | 73 | 269/3 | 648 | 731 |
| 30000 | 10 | 86/10 | 904 | 914 |
| 35000 | 0 | 14/5 | 986 | 986 |

**Table 2. MMTP on Infrared and Rangelan**

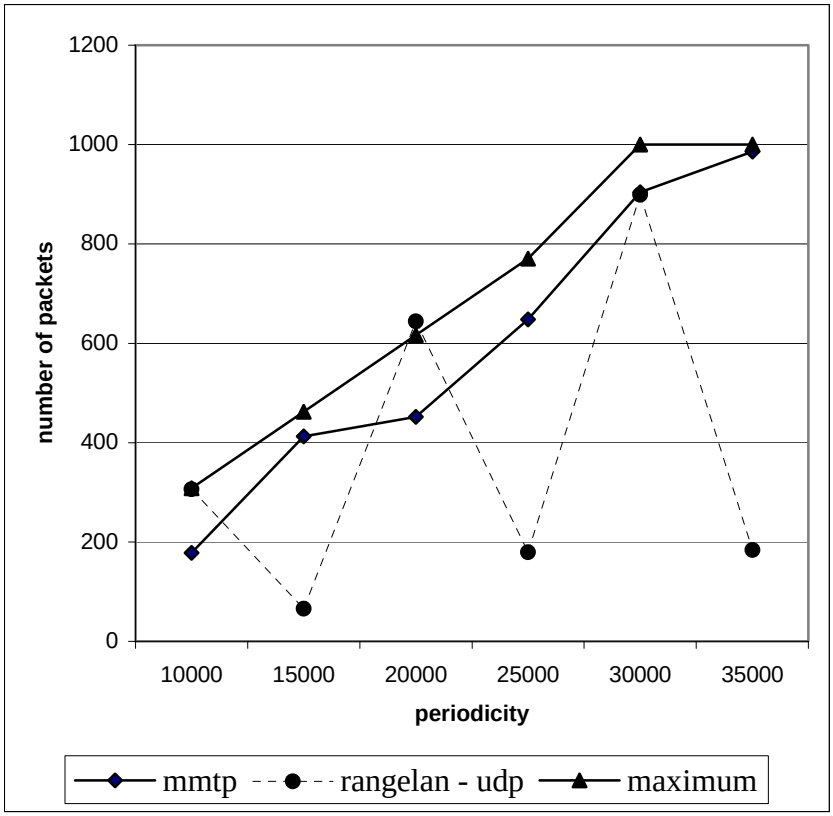



**Figure 42. Comparison of UDP, MMTP and Theoretical Maximum for Packets versus Periodicity**

The current version of MMTP is too conservative on the low range, dropping too many packets, and seems to perform worse than the simple CBR flow at certain periodicities. However, when comparing the performance of MMTP and simple UDP CBR, it must be pointed out that UDP is flooding the link, wasting resources both on the wired and on the wireless link. The high-speed links of the University infrastructure mask this effect on the wired infrastructure, and allow the CBR flow to course through the network until packets are dropped on the wireless base-stations for lack of bandwidth. This would not happen on the Internet at large, because lower bandwidth links may be found on the path. MMTP drops the packets at the source, so only packets that are expected to arrive on time are

actually sent. Figure 42 below shows the number of wasted frames. Those frames were received after their deadline had expired.

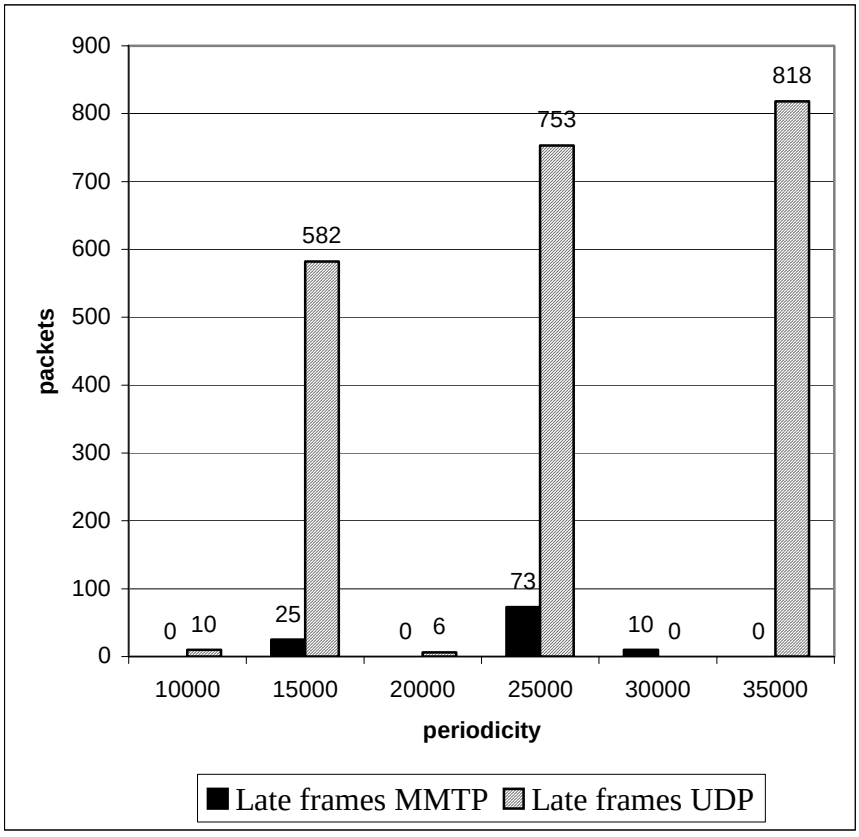



**Figure 43. Packets Received after their Deadline**

Other than losses caused by errors on media, when the periodicity hits 35000 microseconds the joint channel created by MMTP has enough bandwidth to carry all traffic. This is not true to any of the channels taken singly. That means that even in the current form MMTP allows a better quality stream with more reliability than any one of the channels used alone

## 5.5 Conclusions and Future Research

The main contribution of this chapter is the description of MMTP, together with experiments that highlight the advantages of the prioritization scheme for timely delivery of multimedia data. The prioritization scheme is especially important when not enough bandwidth is available to deliver the whole stream within its deadline, and the experiments show how MMTP is able to get more frames delivered in time.

MMTP is a protocol specialized for the transmission of multimedia traffic in mobile, wireless environments. This is both its strength and weakness, in the sense that it cannot be used as a general-purpose protocol (such as TCP and UDP), but it makes building multimedia applications easier by offering a complete solution for flow and congestion control, priorities and even partial reliability through retransmissions. In contrast, building a multimedia application using RTP, for instance, requires extensive work, because no support other than framing is given to the application. Additionally, because congestion control is left to the application, it may lead to the creation of "bad network neighbors", applications that flood the network with data, and do not back off from congestion in a manner that is agreeable to other protocols.

Other strengths of MMTP are the ability of using multiple data-link layers simultaneously, its independence from a single IP address, and its use of the loss discrimination heuristic described in Chapter 4. This allows bandwidth aggregation and mobility, and for a good throughput even in the presence of transmission errors. The

experimental data for bandwidth aggregation and loss discrimination are shown for XMTP in Chapter 4 and for R-MTP in Chapter 6, the last of which shares with MMTP the not only the loss discrimination heuristic but also the multiplexing characteristics.

MMTP was built as a user-level program, and it is limited to the low timer granularity available for user programs. It would be very useful to build it as a kernel module for Linux, to be able to use newer higher speed wireless interfaces. The current implementation is limited to bandwidths of 15Mbits, due to the limitation of a one-millisecond minimum timer interval for user programs in Linux.

Even though MMTP is very specialized, the applicability of MMTP is growing with the proliferation of video delivery to mobile devices. Further development and testing of MMTP is planned, and a video streaming application is the logical next step [TE005].

# Chapter 6 R-MTP

## *6.1 Introduction*

This Chapter presents the design and implementation of a reliable transport protocol with support for mobility using the congestion control and loss discrimination mechanisms described in the previous Chapters. The Reliable Multiplexing Transport Protocol (R-MTP [MA101] [MA201]) is a protocol for the transfer of bulk data[17] in wireless environments, to mobile systems that have access to multiple link-layer technologies. R-MTP is designed as a multiple channel, rate-based protocol that uses selective acknowledgements for reliability, and bandwidth estimation for flow and congestion control.

The use of R-MTP preserves end-to-end semantics, transparently providing the application with the simultaneous use of multiple channels by distributing data from the application across a set of available channels, using inverse multiplexing. This approach exposes link-layer connectivity and per channel resource information to the transport layer, allowing R-MTP to adapt to both changes in available bandwidth on each channel and changes in availability of channels. As the available channel resources change, R-MTP adapts, changing the fraction of flow that is being sent on each channel and adding or removing channels as necessary. For each channel, R-MTP monitors delay and

[17] such as file transfers currently done over TCP by FTP or HTTP

interarrival time for use with bandwidth estimation and congestion control. In addition, R-MTP uses information generated by its rate-based mechanism (interarrival time) for the classification of losses (i.e. congestion losses vs. transmission errors) on the wireless link.

The main contribution of this Chapter is the aggregation of resources across multiple heterogeneous channels in a mobile environment. Such channel aggregation promoted by R-MTP provides four key benefits. First, there is the benefit of a fatter pipe, which enables shorter transmission times. Given that the bottleneck for most wireless communication is the last hop between the mobile host and the base station, the addition of bandwidth at this last hop will alleviate some of the bandwidth constraints on the mobile node. Second, R-MTP's simultaneous use of multiple channels makes it less sensitive to minor bandwidth fluctuations on any one individual channel. This allows for continuous communication even if a "blackout" makes one channel unavailable for a short period of time. Third, by potentially using all available channels, the user does not have to guess which channel is the currently the best. Raw bandwidth alone does not translate directly in higher performance in wireless shared media. A channel with higher raw bandwidth may have lower available bandwidth than a channel with lower raw bandwidth, e.g. due to heavy usage on the faster channel. Finally, smooth vertical handoffs for active data streams are a natural benefit of using multiple channels and of hiding link-layer connectivity. A connection is not switched from one point to another. As the user moves, channels are added and deleted. With enough overlap and slow fades,

the aggregated channel appears continuous.

The remainder of this Chapter is organized into five Sections. Section 6.2 presents the design challenges and related research for the design of a multiplexing protocol in a mobile environment. R-MTP is described in Section 6.3, where the channel abstraction is presented together with the mechanisms for bandwidth estimation, reliability, flow control and channel management. Experimental results are discussed in Section 6.4. The Chapter ends with conclusions and future research in Section 6.5.

## *6.2 Design Decisions and Related Research*

Our approach to supporting reliable communication in mobile environments combines results from three primary areas in computer communications: channel multiplexing, mobility support and rate-based congestion and flow control. In this Section, we discuss the various uses of multiplexing in different environments, together with how multiplexing data into heterogeneous communication channels affects the performance of a protocol. Multiplexing naturally uncouples the transport protocol from the network layer, and allows a simple path for mobility. Successful load balancing across multiple channels requires knowledge about the available bandwidth of every communication channel. The use of rate-based mechanisms for transmission allows precise control over how much data is sent on each communication channel. The Homeostatic Congestion Controller, presented in Chapter 3, measures bandwidth and allows for fine-grained load balancing. The loss discrimination heuristic presented in Chapter 4 diminishes the impact

of transmission errors introduced by lossy wireless channels in the throughput of the reliable transport protocol. We also present the advantages of path diversity for better throughput.

### 6.2.1 Communication Channel Multiplexing

With the growing popularity of wireless access, coverage areas are growing and often overlapping, enabling more than one communication technology in a specific area. The combination of hosts with multiple interfaces and this overlap of coverage enable the simultaneous use of multiple channels. This adds a source of additional bandwidth to mobile networking that has not yet been tapped. Current channel aggregation techniques are limited to identical channels across a homogeneous link layer. Our solution enables the simultaneous use of channels with varying characteristics (e.g. bandwidth, delay). Figure 44 depicts a typical scenario where a mobile can use three link layer technologies at the same time.

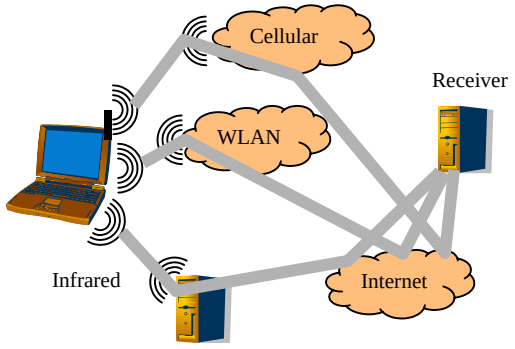



**Figure 44: Example scenario with infrared, wireless Ethernet and a cellular modem**

The simultaneous use of communication channels from multiple technologies enhances

communication support for mobile devices in several ways. The first advantage is the obvious increase in bandwidth through bandwidth aggregation across channels. In traditional wired communication, the bottleneck link to a host is often some router in the network and additional bandwidth at the endhost will not alleviate the problem. In wireless environments, the bottleneck link is often the last hop. By expanding this last hop to the mobile host to encompass multiple channels combined to form a single virtual channel, we can increase the capacity of the bottleneck. The second advantage is to free the system from committing to a single choice of a link-layer channel. By providing information about currently available channels, the system can choose some subset of these channels for its communication. Using every available link layer all the time may not necessarily be the best solution, since the addition of some resource poor channels may actually hurt performance. The choice of which channels to use depends on the constraints set by the user (e.g. cost), the protocol (e.g. performance) and the operating system (e.g. power consumption). On the other hand, the use of all channels, even a resource poor one, may allow communication to flow in the event of a blackout or congestion in any one of the link-layers currently being used.

The aggregation of multiple communication channels has been attempted at different layers of the protocol stack. At the link layer, such aggregation for serial devices has been implemented in both Linux and Windows and many commercial routers. In both operating systems, the characteristics of all communication channels must be the same and a load-balancing algorithm is used for scheduling transmissions. Linux, Cisco IOS and Sun also allow for the use of multiple Ethernet adaptors in the same fashion, in what

is called respectively “bonding”, “etherchannel” and “trunking”. This type of channel aggregation has to be done between the same two endpoints because this is a link layer solution, and transparent to the upper level protocols. The novel aspect our multiplexing solution is that we expose multiplexing to the transport level. The base-stations or access points are not aware that a virtual pipe aggregating many channels is being created, and therefore they do not have to be modified in any way. Only the hosts at the end-points have to be aware of the multiplexing, thus providing an end-to-end solution. This allows the use of heterogeneous technologies with diverse attachment points.

ATM also allows for channel aggregation. The aggregation of many lower bandwidth channels in a larger pipe is called "reverse multiplexing" [CH098], and is now part of the ATM specification. The basic idea is to provide a variety of rates and flexibility in allocating bandwidth for commercial services. Since ATM depends on in-order delivery for cell reassembly, much effort has been put into maintaining the strict synchronization of different flows. IP does not have any ordering requirements, but in general transport protocols will suffer with channel reordering. By implementing multiplexing at the transport layer, such strict timing and ordering requirements can be overcome by buffering in the endhost if necessary. Similar work has also been done in the aggregation of bandwidth [SN099] in wireless links using the facilities of PPP (multilink) [SK096]. The mechanisms described in this Chapter are more general, allowing the use of any interface that supports IP.

The addition of multiplexing at the transport layer introduces an increase in the

occurrence of out-of-order delivery of packets due to the different delays experienced on the different channels. Large differences in delay have two effects on multiplexing transport protocols. First, a packet being transmitted over a channel with long delay may appear to be lost if the loss detection methods do not consider the channel being used and the characteristics of that channel. Gap detection can be used on a per channel basis, but has no meaning across channels. Similarly, the use of duplicate acknowledgements for quick notification of lost messages is meaningless if the sending channel is not taken into account. Second, large differences in delays across channels will affect the performance of the protocol. Most applications in current use require not only a reliable protocol, but also a protocol that provides in order delivery. Application-level framing [CL090] may free protocols from providing resequencing and allow applications to deal with out-of-order reception, enabling the faster use of incomplete data. In order to provide a standard reliable in-sequence transport service for existing applications, data has to be reordered. Transmission across channels with different propagation delays, as well as standard loss and reordering on a single channel, causes out-of-order delivery. Reordering can be accomplished by buffering packets that arrive early until the preceding packets have all arrived. The amount of out-of-order delivery that can be handled by a protocol can be managed by limiting the size of the receiver's buffer. The minimum buffer size should be larger than twice the aggregated delay bandwidth product of all participating channels, so all channels can send data at their maximum rate without running out of buffer space for reordering. Because channels may be added and deleted during the lifetime of a connection, this value should be chosen conservatively.

The use of multiple link layers introduces the concept of diversity. There are two nice side effects of using multiple link layers. The first is communication diversity at the link level. That means that transmission errors on the interfaces may be uncoupled (a jamming signal on the 2GHz band would not affect the interface at 5GHz). The second is possible path diversity, if the attachment points are in unrelated positions (for example, a cellular phone carrier network and a campus network). The end-to-end path will probably have unrelated congestion events. This conspires for a better general throughput. In [PU004], a monitoring architecture is created to attain the same objectives as the Homeostatic Congestion Controller, to try to achieve diversity and bandwidth aggregation.

## *6.3 R-MTP*

Network protocols must be improved to cope with changes and advances in wireless and mobile technology. The design of a reliable protocol for wireless communication faces various challenges. First, wireless channels often provide much lower bandwidth than their wired counterparts. Second, inherent in the use of wireless communication is the effect of movement of the wireless device. Third, the lossy nature of wireless channels degrades the performance of any reliable transport protocol optimized for the congestion-based losses seen in wired environments. In response to these challenges, we have designed R-MTP, a reliable transport protocol that (1) aggregates the bandwidth from multiple channels to overcome bandwidth limitations, (2) provides mobility support at the transport layer to overcome the limitations of network level solutions, and (3) uses rate-based flow control to aid in the distinction between transmission-based and congestion-

based losses.

In this Section, the architecture of the *channel* abstraction for R-MTP is presented together with the parameters used per channel and across multiple channels. Then the mechanisms for reliability and flow control are analyzed. Finally, the frame format for R-MTP headers are presented. R-MTP uses the Homeostatic Congestion Controller described on Chapter 3 for bandwidth estimation on each channel, and the loss discrimination heuristic described on Chapter 4 to get better performance on wireless links.

### 6.3.1 Protocol Architecture and Parameters

Our approach defines the architecture of an end-to-end transport layer that supports the aggregation of resources from multiple end-to-end network layer channels. The transport layer provides a multiplexing service that maps a single virtual application layer channel across these multiple network layer channels. The application submits data units to the end-to-end transport layer for transmission; the transport layer processes them and interacts with the various network channels to determine which channel to use for transmission. The transport layer on the receiving side processes the incoming data units from all channels and presents them to the application layer in a meaningful way, where meaningful is defined by the specific application requirements.

Our goal is to design the transport layer to adapt its behavior to changes in the availability of individual link-layer channels, as well as the characteristics of the specific channels to

which the transport layer has access. On startup, a collection of one-way network channels is established and each of these channels is probed via the packet pair method [KE092] to determine available bandwidth on the channel. This technique provides the current value of the smallest interval at which packets can be sent without experiencing queueing delays. Delay, or round trip time, is measured by recording the time a frame was sent and its acknowledgment received. These measurements are used to define the main parameters of R-MTP, which are used by its reliability, congestion control and flow control mechanisms.

For reliability and sequencing, R-MTP uses a window-based reliability mechanism with selective acknowledgements. The size of the reliability window, *window size*, defines how much space is available at the receiver for active messages, where an active message is a message that may still be accepted and buffered by the receiver. The value of *window size* depends on the bandwidth and delay estimates for all channels currently in use. Since a fixed size bitmap is used to implement the selective acknowledgements, the window size is fixed during the lifetime of the connection, to maintain a constant header size. Processing is simplified if the header is word aligned, therefore, the *window size* is defined as the minimum multiple of 32 that is greater than twice the sum of the bandwidth delay product of each channel. The bandwidth delay product is the number of packets in flight at each time. The reliability window must accommodate enough packets so protocol processing does not stop if a packet is lost. This requires the window to be large enough to accommodate all the packets transmitted from the time a packet was lost

until a replacement arrives.

R-MTP tracks the available buffer space at the receiver with the variable *free buffer space*. The receiver maintains a receive buffer, equal in size to the reliability window, for buffering messages that have already been processed by R-MTP, but not yet passed up to the application. The amount of free space in this buffer dictates how far the sender can slide its send window upon receipt of acknowledgements from the receiver. If there are no more free buffers, R-MTP can continue processing messages in its reliability window, but will not be able to advance its reliability window. R-MTP uses a field in its header to report the value of *free buffer space* to the other side of the protocol. R-MTP also maintains the parameter *maximum receiver rate* to track the maximum rate at which a receiver can process frames. If transmission is limited by the processing power of a receiver, R-MTP caps its transmission rate at *maximum receiver rate*.

Another variable is used to control the maximum amount of data R-MTP sends through a single channel. R-MTP tracks the maximum channel bandwidth per channel both to avoid trying to send too much data across a channel and to make sure a channel is used to its capacity. Since R-MTP is a rate-based protocol, R-MTP translates this information into a *maximum rate* that is sustainable on the channel. To use all available bandwidth, R-MTP continuously probes active communication channels. Probing may cause packet loss if it is done when a channel is operating at its limit, since probing beyond the maximum available bandwidth will momentarily exceed that maximum. If the maximum channel bandwidth is known, and the *maximum rate* is set to that value, R-MTP will stop

probing when the maximum bandwidth of a channel is achieved.

The ideal rate is an elusive target. It should be as close as the maximum allowed by the channel before causing congestion, but since channels are shared, the ideal rate will change over time. To start the process, an initial rate needs to be defined. The interarrival time measured at startup is used to calculate the *initial rate*. The ideal rate is the inverse of the minimum period that the channel can support without causing congestion, which is given by the interarrival time measured by the packet pair method. Since this measurement may contain errors, the *initial rate* is defined as half the calculated rate. The rate will track the available bandwidth, and eventually converge to the inverse of the interarrival time or the *maximum rate* defined above.

### 6.3.2 Reliability

R-MTP uses retransmission-based reliability and gap-detection for identifying losses. The sender is notified that frames have arrived at the receiver by receiving acknowledgements. There are two types of acknowledgments: cumulative and selective. The cumulative acknowledgment carries the number of the next expected frame (last received + 1). The selective acknowledgment is a bit map of the state of the received queue, with bit 0 being the position of the next expected frame. The sender keeps an individual bit map for each channel to record which frames were sent on the channel.

Gap detection works by checking the selective acknowledgement for gaps in the sequence of each channel. Every time a frame is sent through a channel, it is marked in

that channel's bitmap. When the selective acknowledgement is received, it is compared with each channel bitmap. If gaps are found, the missing frames are retransmitted. Once a frame is retransmitted, the protocol can no longer rely on sequencing to detect gaps, since the retransmitted frame is out of order. Two additional global bit maps are kept to track information about retransmissions. One bitmap, *retransmitted frames*, tracks all retransmitted frames to prevent multiple retransmissions of the same packet before it has had a chance to be successfully received and acknowledged. In order to determine when a retransmission has been lost, R-MTP maintains a second bit map for *marker frames* that is used to indicate the frame that was sent immediately after a retransmission. If an acknowledgement for the *marker frame* arrives before the acknowledgement for the retransmitted frame the retransmitted frame is deemed lost, and it is sent again.

*Marker frames* are needed because even after a gap is found and a lost frame retransmitted, all acknowledgements generated before the lost frame arrives at the receiver will contain the same gap. If a frame were retransmitted every time a gap was detected, there would be many duplicated retransmissions. To prevent unnecessary retransmissions, a frame in the *retransmitted frames* bitmap can only be retransmitted again after its marker frame in the *marker frames* bitmap has been cleared. The position in the *marker frames* bitmap is cleared when an acknowledgement or a gap for that position is found. This allows the protocol to do all its retransmissions without using explicit timers.

Consider the case depicted in Figure 46, of a channel that transmits frames 3 and 5, and

receives an acknowledgement for 5. In this case, frame 3 is deemed lost. Frame 3 will be retransmitted by the first available channel, cleared from the original channel bitmap and set in the bitmap of the channel responsible for the retransmission. It will also be recorded in the *retransmitted frames* bitmap, and the frame following it in that channel will be recorded in the *marker frames* bitmap. Once a frame is retransmitted, it will only be retransmitted if its marker frame is acknowledged. If frame 14 follows the retransmission of frame 3 in this example, frame 3 can only be retransmitted after the acknowledgment for frame 14 is received. Frame 3 precedes frame 14 in the channel, and if the acknowledgement for frame 14 arrives before the acknowledgment for frame 3, then frame 3 was lost again.

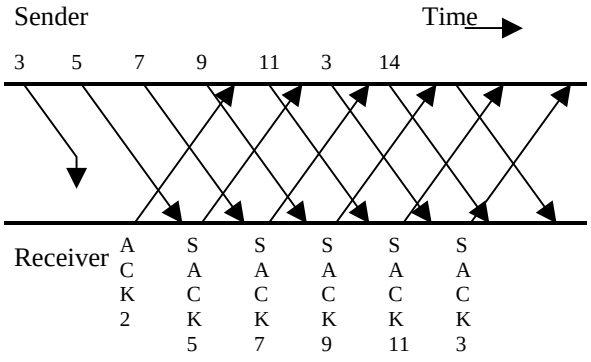



**Figure 45: Detection and retransmission of a lost frame**

R-MTP has no protection against communication channels that reorders frames. Frames that arrive out-of-order in a single channel will cause unnecessary retransmissions. The optimization that folds two acknowledgements into one, to save bandwidth if the data traffic is one way, has a side effect of masking half of single packet reordering, cutting

the number of unnecessary retransmission in half, but do not solve the problem. In addition, the timing fluctuations of reordering channels will adversely affect the rest of the protocol processing, because R-MTP expects channels that behave regularly in the time dimension.

Gap detection may fail if all packets in the *reliability window* are lost. Instead of using an explicit timeout, if the *reliability window* becomes full, and no frames are available for retransmission in the channel's queue, then the rates are halved across all channels and frames are replayed, starting with the oldest unacked frame. This works as a natural timeout mechanism. The rates are halved to prevent a sudden mismatch of window size and bandwidth-delay product to cause many unnecessary retransmissions.

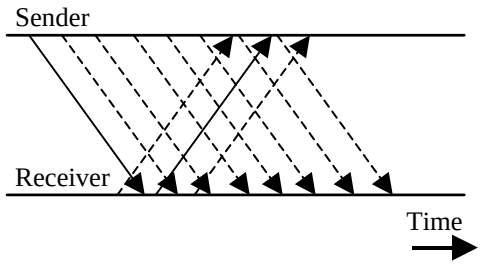



**Figure 46: Bandwidth-delay product in a rate-based environment**

If only one channel is being used, reaching a full reliability window may be caused by underestimating the reliability window size. The minimum reliability window size must be greater than twice the propagation delay times the channel rate, plus an extra period to account for synchronization. In Figure 46, we have an example of a channel that is capable of sending 7 frames before an acknowledgement is received. In this case, the

minimum size of the reliability window is 8. If a smaller size were chosen, every frame would be transmitted twice.

The reliability window has to buffer enough frames to allow the acknowledgement for a frame to be received in time to prevent its retransmission. If more than one channel is being used, then the Minimum Window Size (MWS) is given by a ratio of the rates and the longest propagation delay. The MWS is defined as the number of buffers needed to accommodate all packets that are transmitted between the time a packet is sent on the channel with longest propagation delay and the time that its acknowledgement is received. The total propagation time is the one-way propagation delay of the slowest channel (packet) plus the one-way propagation delay of the fastest channel (ack). The number of packets transmitted in this interval is in a channel is the total propagation time times the rate of the channel. The total of packets is the sum of the number of packets transmitted in each channel. Therefore, MWS is given by the expression in Equation 2.

$$MWS = \sum_{i} (delay\_slowest\_channel + delay\_fastest\_channel) * rate\_channel(i)$$

**Equation 2: Minimum Window Size**

Channels may be added after the initial negotiation of window size. A new channel has to obey the above requirement or timing mismatches will cause unnecessary retransmission. If a channel that violates the timing requirements on the MWS is added to the collection of available channels, every frame sent on this channel will be retransmitted by one of the other channels. The acknowledgements for the frames sent on the slow channel would not

get back in time to prevent the queue getting full, causing retransmissions.

Setting the reliability window size to a value equal to the minimum window size is not a good policy. If packets are lost on a channel with long propagation delay, this may cause the protocol to work on hiccups, similar to TCP's silly-window syndrome.

### 6.3.3 Flow Control

Flow control should play a small part in the protocol, as one driving assumption is the lack of bandwidth on the last hop. Flow control is used so the sender does not swamp the receiver with frames, which assumes that it is possible for the network to be faster than the internal processing on the receiver. On the other hand, it is easy to create a scenario where the mobile is suddenly well connected, for example, if the user takes the mobile to the office and uses a docking station. So the current connections will suddenly have much more bandwidth than previously, and the sender may swamp the receiver.

There are two mechanisms used for flow control. One is the *maximum rate* negotiated on startup. The *maximum rate* defines a hard limit on the number of packets sent per second – even if the protocol measures that the available bandwidth is greater than it is currently using, it will not surpass the *maximum rate*. When the protocol reaches the *maximum rate* it will stop probing until the rate drops. The second mechanism is the receiver queue. This is an ancillary queue that smoothes the bunching effect that the mismatch of rates and delays generates, and is implemented as a part of the receiver window. The flow control works in the same way as TCP, but with a fixed size queue: the packet header

carries how many slots are left on the receiver queue. The sender will stop sending data frames if the queue gets full, halve the rate and send only acknowledgements until it gets a acks from the receiver that shows there is space again in the receiver queue, when normal processing resumes.

### 6.3.4 Congestion Control

R-MTP uses the Homeostatic Congestion Controller for congestion control. Therefore, it uses three mechanisms:

a) packet pair probing, for bandwidth estimation
b) jitter correction, for congestion avoidance and rate correction
c) multiplicative rate decrease, for congestion control

These mechanisms were explained in Chapter 3. R-MTP also uses the loss discrimination heuristic presented in Chapter 4, which alters the multiplicative decrease. Only losses tagged as congestion losses cause the rate to be halved. Losses tagged as transmission losses do not trigger the congestion control mechanism.

### 6.3.5 Adding and removing channels

Channels may be added after the protocol processing has begun. The process starts when the protocol is notified of the availability of a new interface. The protocol creates a socket and binds to that interface, and sends a *add interface* message in one of the other active interfaces. The other peer creates a socket to take care of that interface, and probes the

new channel to measure bandwidth and delay. It then analyzes this information for mismatches: if the characteristics of the new collection of channels obey the Minimum Window Size requirements described above then the new channel is added to the collection of active channels. If the window size is not large enough to accommodate the new channel, it will not join the collection of active channels if its delay and bandwidth characteristics are worse than those of the current channels. If this channel has better characteristics than other current channels, then other channels may leave the collection of active channels.

A channel that is not active does not carry data frames. It is probed regularly to keep its available bandwidth information current. If other channels became unavailable, it may be added to the collection of active channels and used to carry data.

If the protocol is notified that a channel is no longer available, a *remove interface* message is sent to the peer. The protocol then waits until a *removed* message is received on that channel, when it deletes the socket. Upon receiving a *remove interface* message, the protocol stops sending data on that interface, and sends a *removed* message to end processing.

### 6.3.6 Packet headers

R-MTP's header size is variable, and depends on the size of the selective acknowledgement bitmap. The fields are:

- Port_number: integer
- type: the packet type, byte
- seq: sequence number, integer
- last_sent: the sequence number of the last packet sent on this channel, integer
- rate: the channel rate, double integer
- expected: the accumulated acknowledgement, the number of the last packet received across all channels plus one, integer
- window: receiver free buffer size, int
- ack: the selective acknowledgement, variable
- checksum: integer

The meaning of the bits in the *type* field can be seen below (Table 13):

| Bit | 7 | 6 | 5 | 4 |
|---|---|---|---|---|
| If ON | Add Interface | Remove Interface | Interface Removed | Initial Packet |
| Bit | 3 | 2 | 1 | 0 |
| If ON | End Packet | Probe Packet | Ack Packet | Data Packet |

**Table 13: meaning of the bits in the type field**

Multiple *type* bits may be on at the same time, a probe packet that carries data has type 0x03, a data packet carrying a *add interface message* has type 0x81.

The *last_sent* field allows the receiver to detect if a frame was lost in transit. While this does not play any part in the reliability algorithm, it permits the receiver to maintain the accuracy of the interarrival time estimates, by rejecting periods that contain a lost frame.

There are additional header fields for the *add interface*, *remove interface* and *interface removed* messages. They are:

- IP: IP number of the interface
- Port: port number to be used

## 6.4 Experimental Results

R-MTP provides a reliable transport service using bandwidth aggregation and optimized for wireless environments. The experiments in this section are designed to demonstrate the effectiveness of (1) R-MTP as a reliable transport protocol, (2) the operation of R-MTP in lossy environments, and (3) R-MTP's bandwidth aggregation. Evaluation of a reliable transport protocol must include ideal, isolated environments, as well as shared environments. In the first set of experiments, we address issues related to the performance of R-MTP in lossless environments. The experiments compare the performance of R-MTP to the performance of TCP over various link layer technologies. Since R-MTP is a rate-based protocol that will use all available bandwidth, we also show R-MTP's ability to share link bandwidth both with TCP streams and with other R-MTP streams. In the second set of experiments, we address the last two goals: operation in a lossy environment and bandwidth aggregation. We conclude this Section by

demonstrating the effectiveness of R-MTP's bandwidth aggregation techniques, both in lossless and lossy environments.

### 6.4.1 Test setup

The experiments are run with a user-space implementation of R-MTP where R-MTP data is encapsulated in UDP packets. UDP already offers a *port number* and *checksum* so these variables were taken from R-MTP's header. One major concern in rate-based protocols is how to maintain the regularity of the flow. Linux user space timers are controlled by the kernel's HZ variable. Normally the minimum timer interval is 10 milliseconds, which gives the lower bound to the period. The experiments with infrared were done using this timer granularity, which sufficed due to the low bit rate of the interface. Faster interfaces require a finer resolution, and we recompiled the Linux kernel for a 1-millisecond timer resolution for wireless Ethernet experiments.

For new-generation, faster interfaces, the user-space implementation is not able to achieve sufficient timer resolution. Although no tests were run with faster interfaces that required higher timer resolutions, current kernel implementations (Linux 2.6) allow better timers with nanosecond resolution by using the APIC timer. The APIC timer is used to synchronize processors in multi-processor systems, but it is available as a high-resolution alarm in single processor systems. These timers are available as a patch in earlier kernels (Linux 2.4) and which was developed by Oberle [OB001].

In the experiments, for each run the test program sends one thousand packets of 1400

bytes, using both R-MTP and TCP. The receiving program records the arrival time of each packet, and the graphs plot the cumulative time of arrival of each packet and their interarrival time. The tests cover a sample of wireless link layer technologies: three types of infrared devices and two types of wireless Ethernet. Infrared has two different standards for point-to-point communication: IRLan, which is and IRDA standard, a LAN emulation over infrared, and IRNet, which is not yet an IRDA standard, but is the *de facto* standard for Windows. IRNet uses a leaner stack than IRLan, and should present better performance because it also allows for link compression. IRNet and IRLan have been implemented for Linux. For wireless Ethernet, we used the 900MHz version of WaveLAN, the 2.4GHz Web Gear Aviator PCMCIA card. For wired Ethernet, we used 3Com 574 PCMCIA adapters.

The test setup consists of two of two Sony laptops, a PCG-F360 and a PCG-F450, both running RedHat 6.2 Linux with 2.4.0 kernels. For testing purposes, compression was disabled on the IRNet link, so the packet contents would not influence the results. The IRNet link used the built-in interface of the PCG-F450 and an Actisys 2000+ dongle on the serial port of the other laptop. The IRLan link consists of another Actisys 2000+ dongle on the first laptop and an HP NetBeamIR connected to a 10Mbps hub, and through this to the Ethernet card on the second laptop. Both WaveLAN and Aviator cards are used as point-to-point links.

### 6.4.2 Performance of a single link layer under lossless conditions

The first set of experiments address the desired characteristics of a general-purpose transport protocol. We explore the performance of R-MTP over different link-layer technologies by comparing the arrival time of packets in a sample run with the arrival times of the same sample run using TCP. The capacity of achieving the maximum bandwidth available on a link layer can also be inferred by comparing R-MTP's performance with TCP's. Measuring the interarrival times of packets shows the different ways both protocols achieve this. The last experiments in the first set relate to how R-MTP shares the available bandwidth on a link.

| Link Layer Technology | Total Run Time R-MTP (sec) | Total Run Time TCP (sec) |
|---|---|---|
| IRLan | 159.5 $\pm$ 3 | 160.9 $\pm$2 |
| IRNet | 156.0 $\pm$ 4 | 160.0 $\pm$ 3 |
| Wavelan | 10.69 $\pm$ 0.7 | 9.89 $\pm$ 0.4 |
| Aviator | 8.64 $\pm$ 0.5 | 9.03 $\pm$ 0.6 |

**Table 14: Comparison of R-MTP and TCP over various wireless technologies**

Table 14 presents a comparison of R-MTP and TCP over wireless channels with no external losses. In general, we expected TCP to perform better than R-MTP since R-MTP is implemented in user space. Surprisingly, R-MTP performs better than TCP even when no external losses are introduced. On slower network interfaces, a node's CPU can feed data to the network interface faster than the data can be transmitted on the link. This allows a single transport layer connection to use all available bandwidth of this link.

Under this conditions, TCP's bandwidth probing can cause losses by congesting the link. This results in a net performance below R-MTP's, whose bandwidth probing rarely causes losses in these experiments.

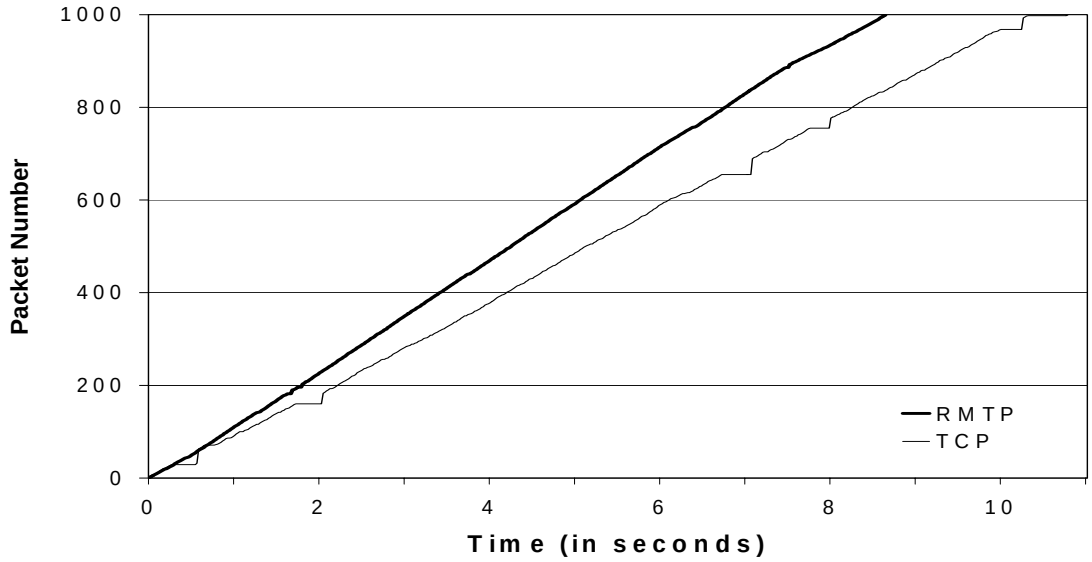



**Figure 47: Packet arrival times for R-MTP and TCP on Aviator**

Figure 47 and Figure 48 show the arrival times of TCP and R-MTP with the Aviator wireless Ethernet and with IRLAN. Packet losses can be observed on the TCP stream by the small horizontal traces that mark a packet that was lost and retransmitted. R-MTP does not show losses due the fact that R-MTP never sends more than the maximum rate over either of the links. The poorer performance of TCP over Aviator is due to TCP's probing mechanism causing more losses on that wireless link. IRLAN, on the other hand, is a reliable medium and losses simply cause short delays in transmission, which do not affect TCP as badly as real packet losses.

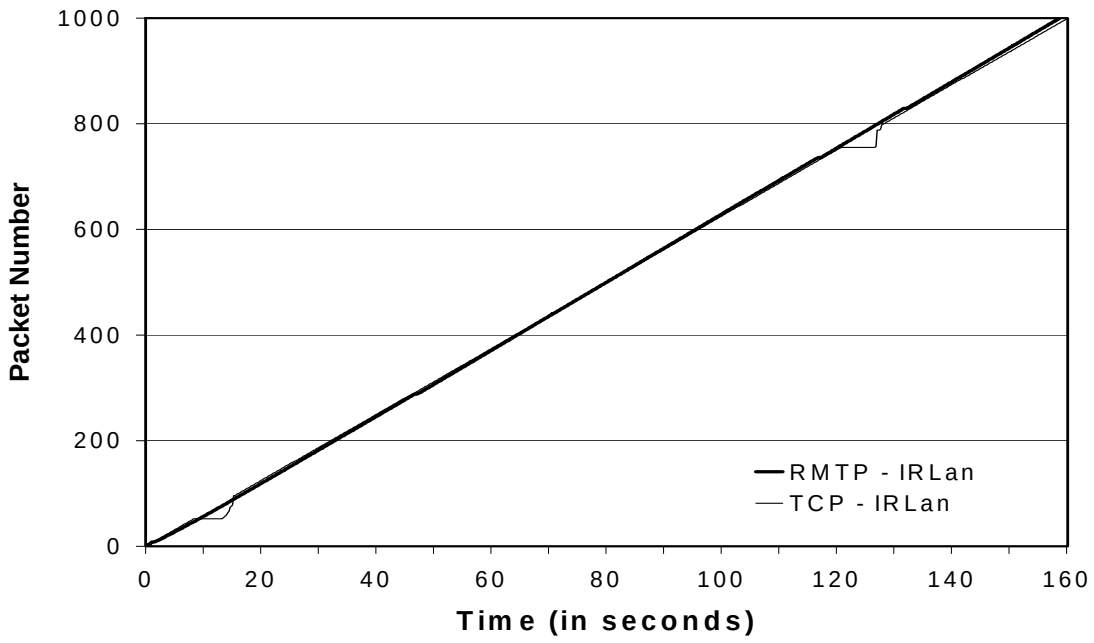



**Figure 48: Packet arrival times for R-MTP and TCP on IRLAN**

The difference in the bandwidth tracking strategies can also be seen in Figure 49, where the plot of interarrival times of TCP and R-MTP is shown. The packet rate chosen on startup by R-MTP is conservative. The staircase represents the convergence of R-MTP's bandwidth estimation to the optimal value. The heartbeat effect on the graph is caused by R-MTP's probing mechanism. Probe packets are sent immediately after a data packet. This minimal interarrival time is the cause of the low points in the graph. The high points are caused by the fact that the next data packet is sent at its regularly scheduled time, two periods after the first data packet, not one period after the probe packet. The interarrival time drops when a packet containing a new rate is received. The rate must be a multiple of the operating system timer *quantum*. If the optimal rate cannot be achieved due to the resolution of operating system timers, the rate will oscillate between values above and

below the optimal rate. This can be seen as the oscillations at the right end of R-MTP's interarrival time line. The operating system timer resolution in this case was 10 milliseconds, and the optimal rate was approximately 145 milliseconds. R-MTP's rate oscillates between 160 and 130 milliseconds. In contrast, TCP's interarrival time oscillates in a much higher range, from 130 to 270 milliseconds, as shown by the trends in the scatter graph representing interarrival times for TCP. TCP sends packets in bursts, causing many packet to arrive back-to-back. For an infrared interface, the reception of each packet takes 130 milliseconds. TCP then waits until its transmission window opens again. At this point, a new burst of packets is sent. Both strategies achieve good bandwidth usage, which can be seen in Figure 48.

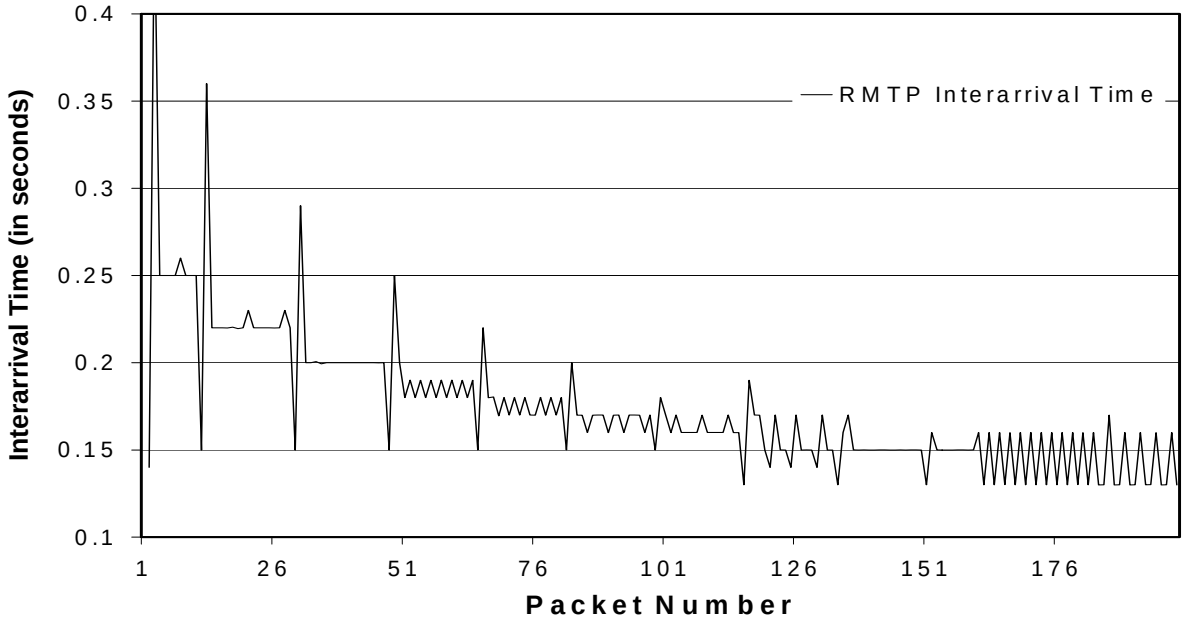



**Figure 49: Packet interarrival times for R-MTP over IRLAN**

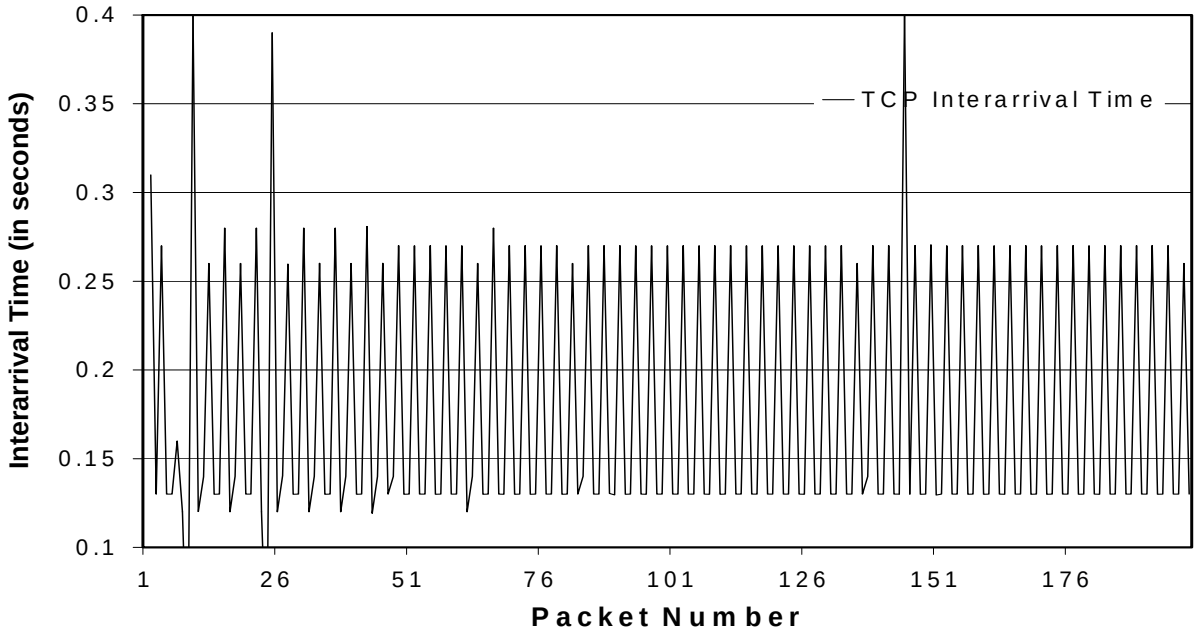



**Figure 50:Packet interarrival times for TCP over IRLAN**

The final experiments in this section demonstrate bandwidth sharing for both TCP and R-MTP. Figure 51 shows two TCP streams running concurrently over the same medium. The stream that starts at a later time is able to acquire enough bandwidth to finish sending its data together with the first stream. This actually translates into a greater bandwidth share than for the first stream. As expected, TCP connections share bandwidth well with other TCP connections, backing off and allowing bandwidth to be divided evenly. Similarly, two R-MTP streams running concurrently can be seen in Figure 52. As soon as the second stream is started, the first stream backs off, marked by the knee in the upper line of Figure 52. During the period that both streams are active simultaneously, the first streams sends 430 packets, while the second stream sends 400 packets. This demonstrates

that R-MTP streams share bandwidth well with other R-MTP streams.

In the final experiment, TCP and R-MTP were run concurrently on the same link. TCP connections are very sensitive at their startup. To measure the influence of available bandwidth on TCP startup, experiments were run with TCP starting on an empty channel and on a channel where R-MTP was using all available bandwidth. Figure 53 shows the graph where TCP starts first. As soon as R-MTP starts, TCP backs off, and drops its rate below that of R-MTP. As the run progresses, TCP increases its rate, and R-MTP backs off, which is seen by the decreasing slope of R-MTP's line. In Figure 54, R-MTP starts first, and backs off after TCP starts. In the period that R-MTP and TCP are active simultaneously, R-MTP sends 300 packets, while TCP sends 250. We expect that TCP's greater sensitivity to losses will allow R-MTP to acquire a greater share of bandwidth, but R-MTP's bandwidth tracking algorithm does not starve out TCP connections.

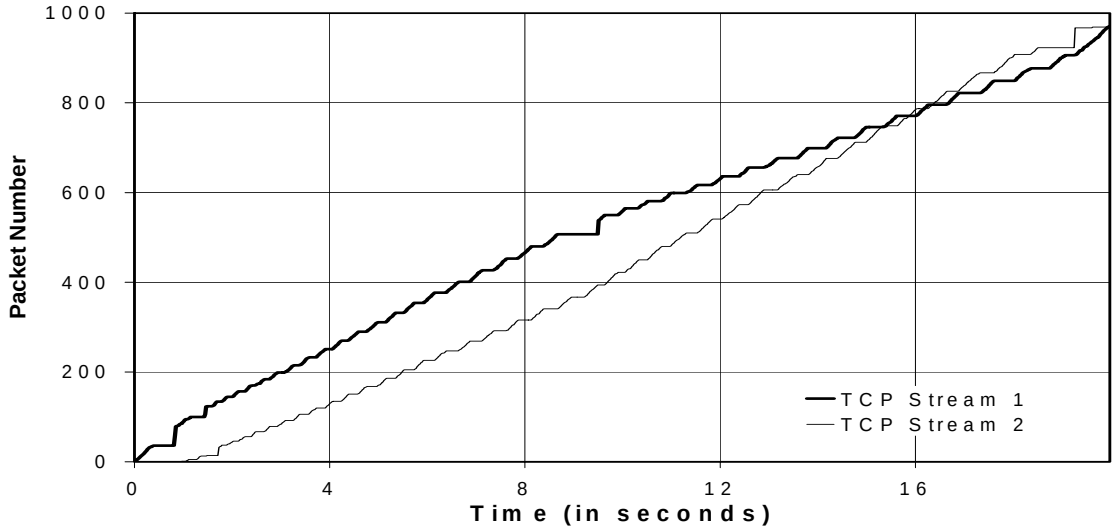



**Figure 51: Bandwidth sharing with two TCP streams**

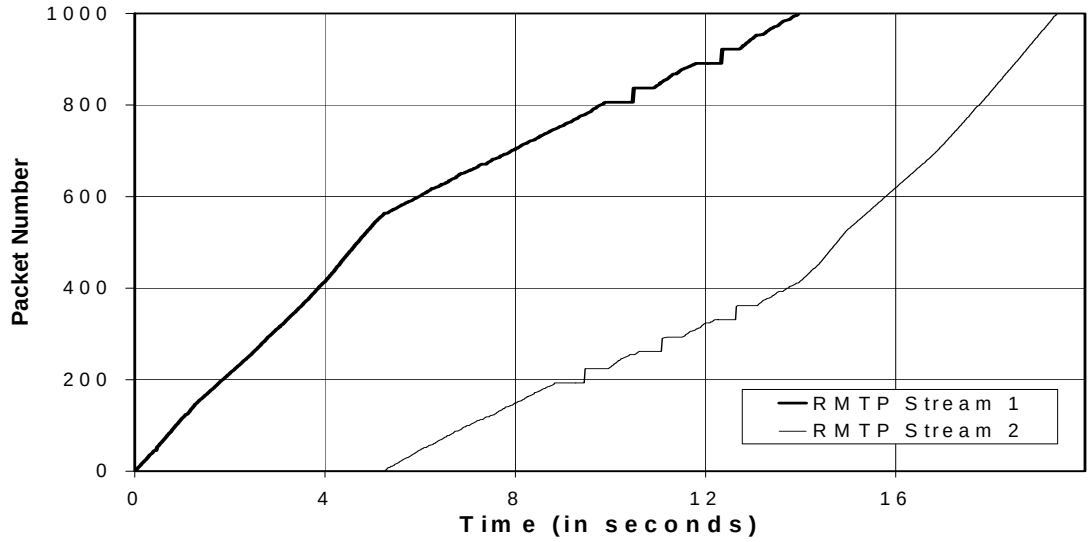



**Figure 52: Bandwidth sharing with two R-MTP streams**

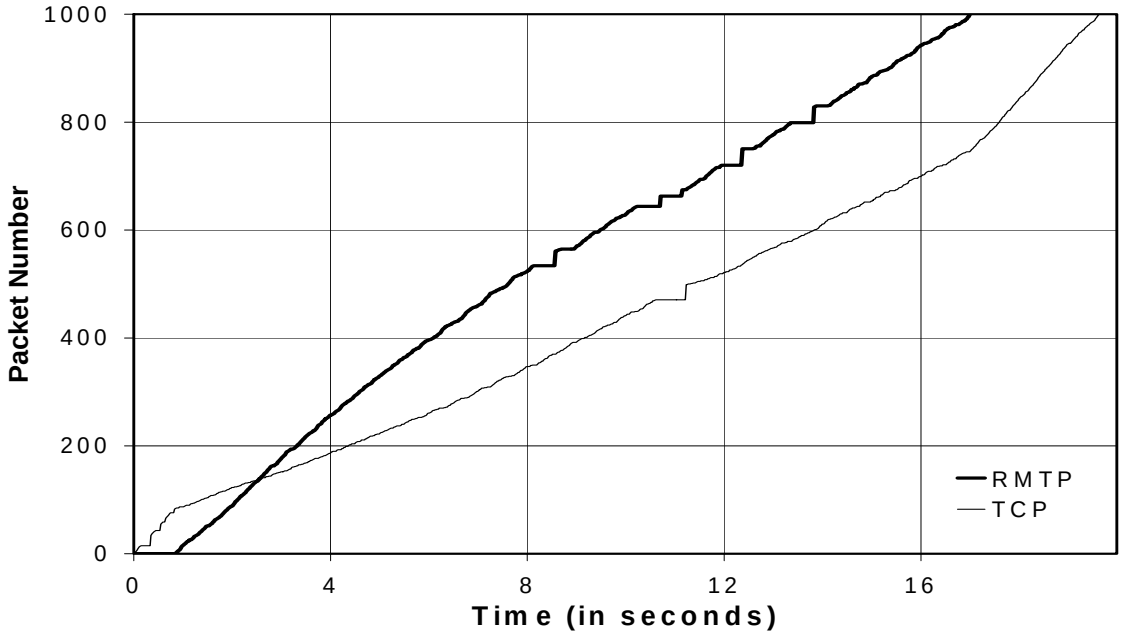



**Figure 53: Bandwidth sharing between R-MTP and TCP (TCP started first)**

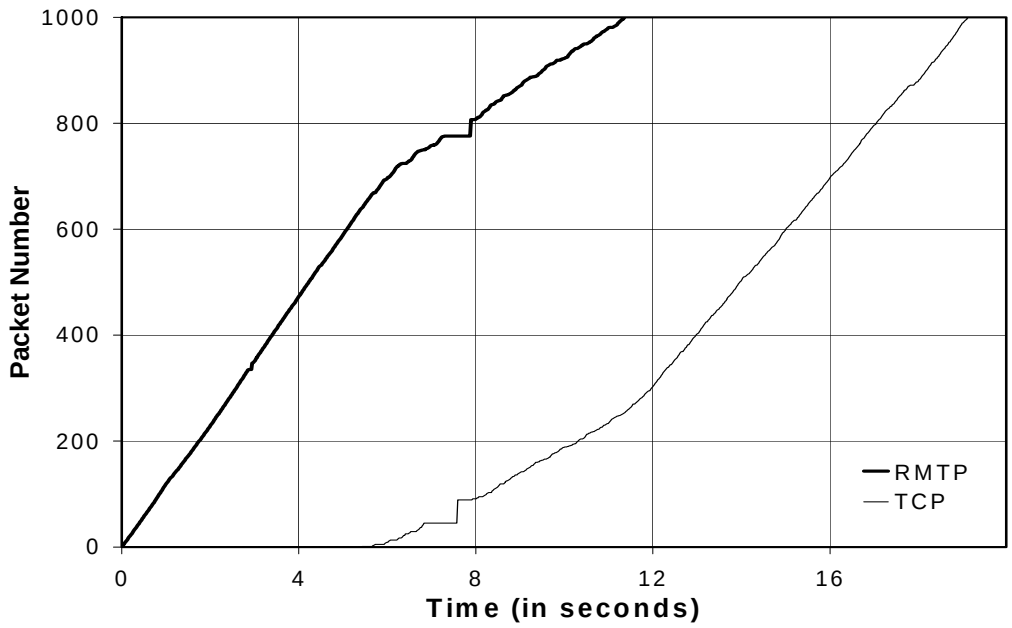



**Figure 54: Bandwidth sharing between R-MTP and TCP (R-MTP started first)**

### 6.4.3 Performance on Lossy Links and Bandwidth Aggregation

The experiments presented in this section address the effect of losses on R-MTP and R-MTP's ability to aggregate bandwidth from multiple channels. The protocol performance on a lossy link is analyzed by comparing the arrival times of packets in R-MTP with the arrival times in TCP. Bandwidth aggregation is shown by comparing the arrival times of packets in a sample run using two link layers with the arrival times of the sample run using each link layer individually. The final experiments in this section demonstrate the validity of using multiple interfaces, by showing how significant advantages may be gained in the presence of losses, even when there is a large bandwidth mismatch on the aggregated interfaces.

A main driving force for the development of R-MTP was performance on lossy links. We expect R-MTP to perform significantly better than TCP on lossy environments due to discrimination of transmission losses from congestion losses. To create lossy environments on our test machines we used the Netfilter framework and iptables libraries to create a user space program that randomly discarded a percentage of arriving packets. This simulates transmission problems where packets are corrupted in transit. The packets still use bandwidth on the wireless medium, but are seen as corrupted and discarded at the receiver.

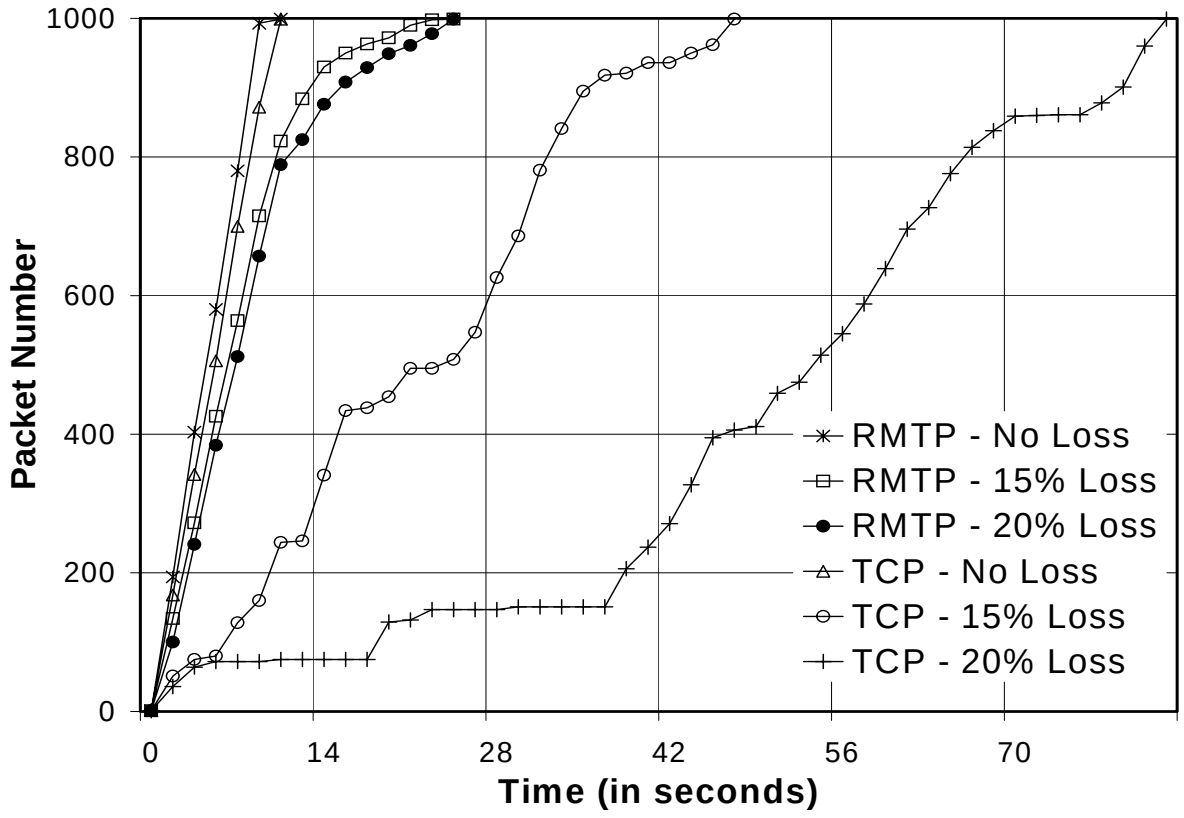



**Figure 55: R-MTP and TCP with no losses and with 15% and 20% losses**

Figure 55 shows a comparison of TCP and R-MTP with different degrees of packet loss. Both TCP and R-MTP coexist very well with losses below 10%. As the number of losses grows, TCP starts to experience more and more slow starts. At 20% loss, TCP takes 10 times longer to deliver the same amount of data than it takes on a lossless link.

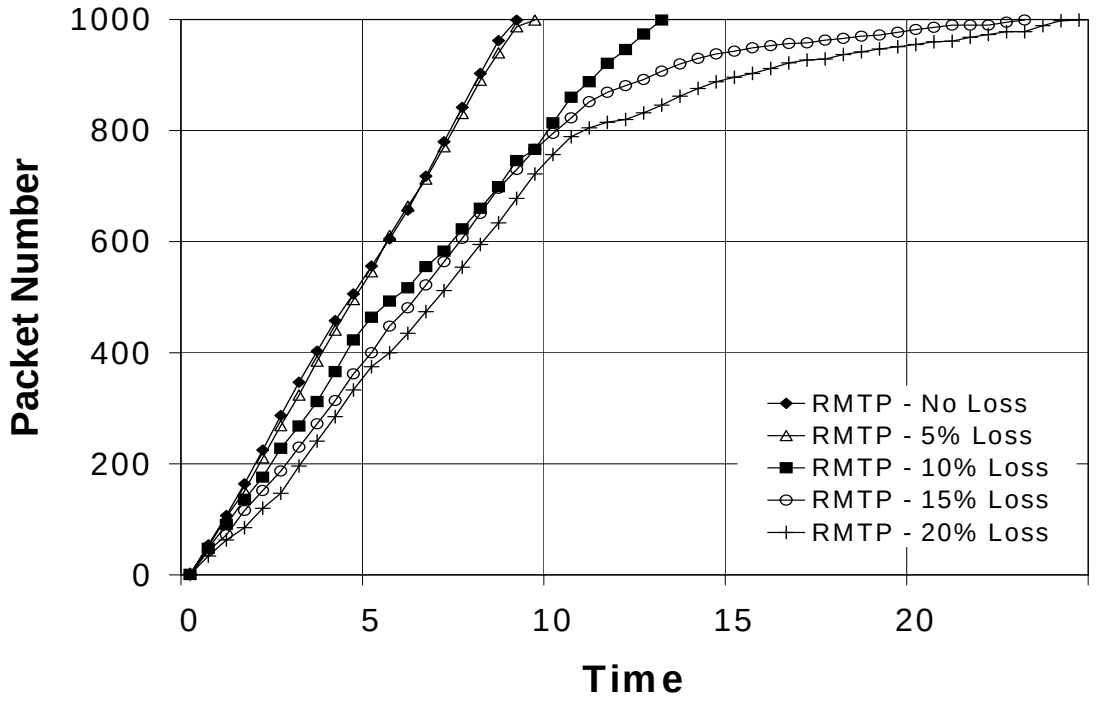



**Figure 56: R-MTP with no losses and with 5%, 10% and 15% and 20% losses**

R-MTP is by design more immune to packet loss. Certain loss patterns may fool R-MTP's congestion avoidance algorithm into confusing transmission losses with congestion losses. This will cause R-MTP to cut back its sending rate. This effect can be seen in the tail end of the transmission at 15% and 20% losses in Figure 56, which shows

the effect losses on R-MTP in more detail. Losses that are miscategorized by R-MTP slow the probing mechanism, hindering recovery, and may prevent probing from working, if the probe packet or the preceding packet is lost.

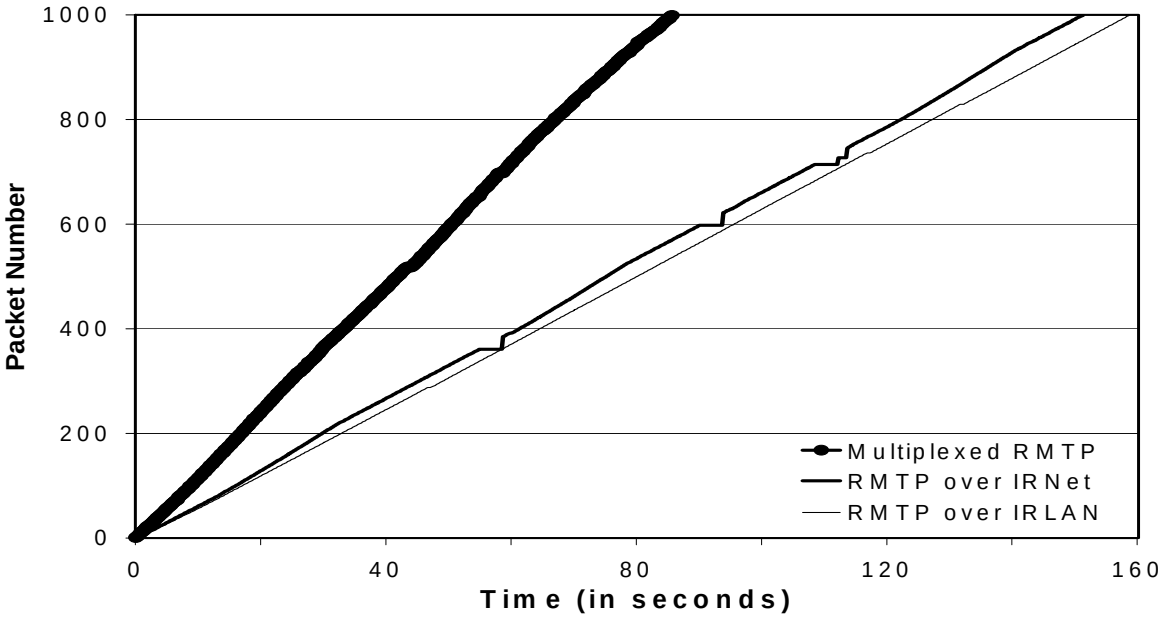



**Figure 57: Bandwidth aggregation on IRNet and IRLAN**

Dealing with losses is a good argument for using multiple link layers simultaneously, even when bandwidth aggregation does not seem a compelling argument. Bandwidth aggregation may significantly increase the total bandwidth available to a mobile host if the link layers have similar capacities. So aggregating two 115Kbps channels is a good choice. On the other hand, it is not intuitive that aggregating a 1.2 Mbps and an 115Kbps channel will add enough bandwidth to offset the cost of multiplexing. The result of both bandwidth aggregations can be seen in Figure 57 and Figure 58 below. Figure 57 shows the aggregation of two infrared channels at 115Kbps, an IRLAN channel and a IRNET

channel. The aggregate channel is considerably faster than each individual channel. This is not the case for the example shown in Figure 58, which shows that the channel resulting from the aggregation of an Aviator wireless Ethernet and an infrared IRLAN connection is only slightly faster than the individual Ethernet channel. In fact, the multiplexing overhead makes the composite channel slower at some points.

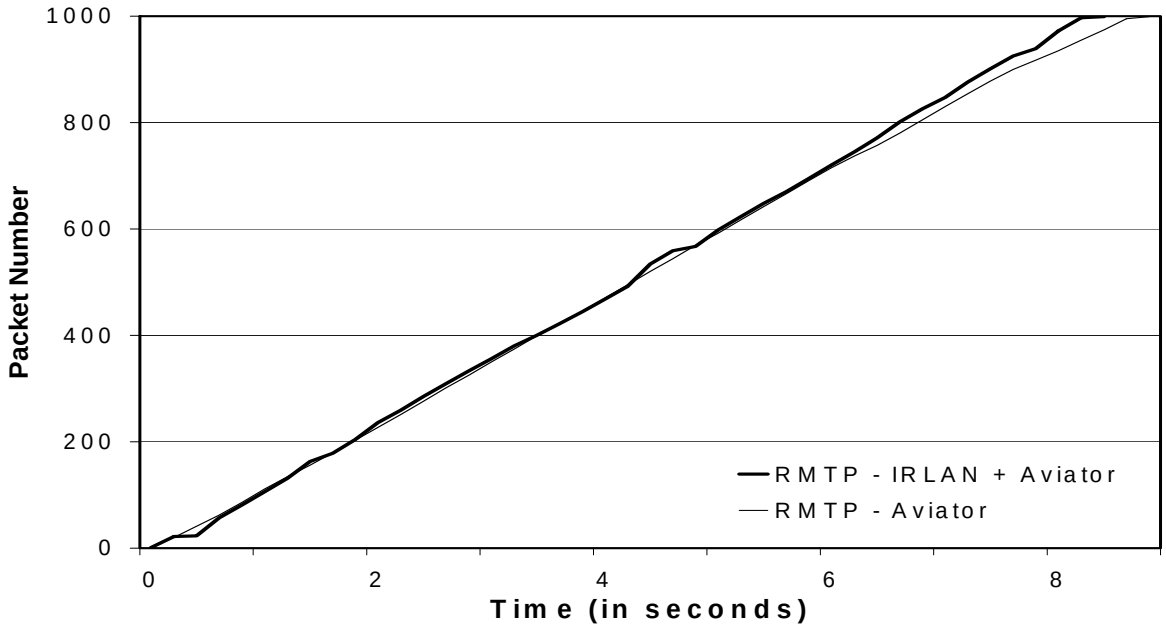



**Figure 58: Bandwidth aggregation on Aviator and IRLAN**

Bandwidth aggregation with such mismatching channel characteristics can be justified in certain scenarios. Figure 59 shows an experiment with the same infrared interface and wireless Ethernet of Figure 58, but with a 30% loss rate on the Ethernet. A TCP run, which normally takes 10 seconds to complete on the wireless Ethernet link, now takes 25 times longer. Suddenly the IRLAN link, lossless in this example, is performing better than the Ethernet link, and completes the run in 160 seconds. The third line in Figure 59

is the multiplexed IRLAN/Ethernet R-MTP connection, which finishes the run in 61.2 seconds. Figure 60 shows the same choices for R-MTP. While R-MTP has better behavior on the lossy link, finishing the run on the wireless Ethernet alone in 81.5 seconds, adding the much lower bandwidth link actually improves the performance by 25% in this case.

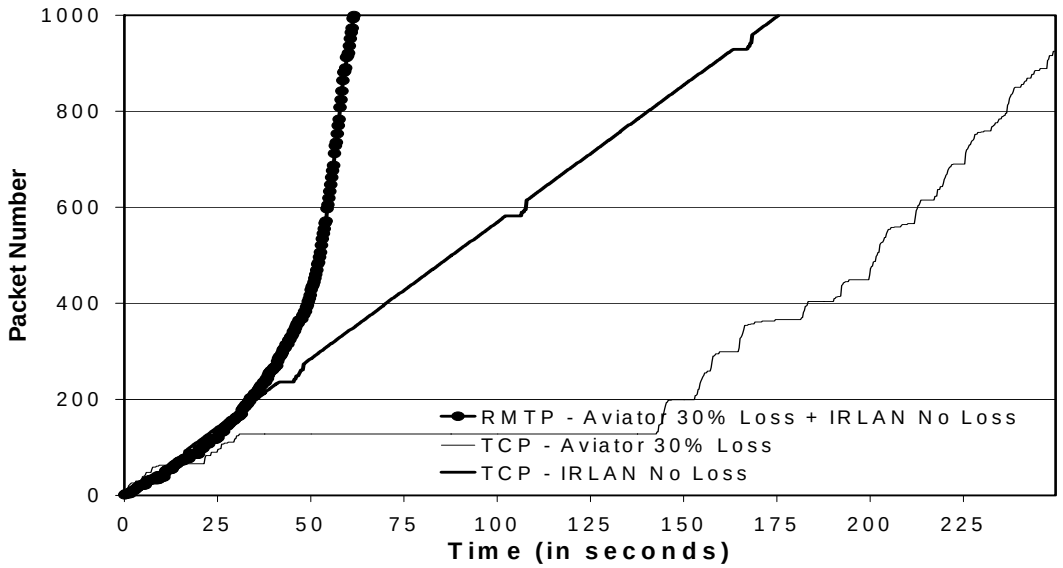



**Figure 59: Comparison of TCP over Aviator with 30% loss and over IRLAN with no loss to R-MTP multiplexed over the same two channels**

Beyond the significant improvement in performance by using two seemingly mismatched interfaces, these results make a case for using all available interfaces all the time. Because it is impossible to measure a priory if an interface is lossy, or to predict noise, using all communication resources that are available creates a virtual channel that is much more resilient than any channel alone. Of course this has to be offset by power and cost considerations, but this opens the door for high availability mobile systems using

current technology.

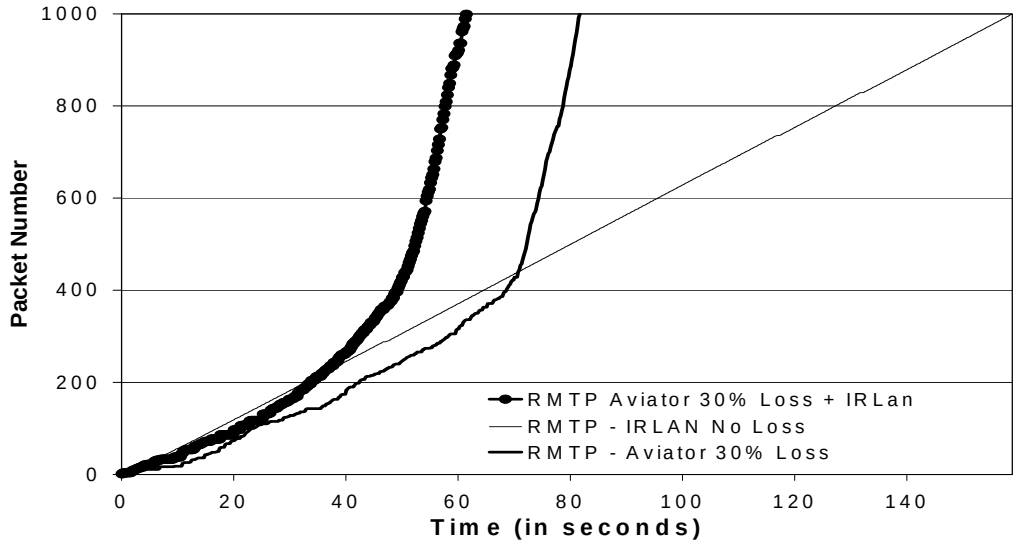



**Figure 60: Comparison of R-MTP over Aviator with 30% loss and over IRLAN with no loss to R-MTP multiplexed over the same two channels**

## *6.5 Conclusions and Future Research*

In this Chapter, we presented R-MTP, a reliable transport protocol designed for mobility and discussed its architecture and design decisions. The main advantage of using R-MTP is its ability of using multiple interfaces at the same time. If a mobile host is in a location where it has access to multiple communication technologies, R-MTP will allow a single application to send its data through all available interfaces, dividing the flow of data transparently among the multiple interfaces according to the measured end-to-end available bandwidth.

The capacity of using multiple interfaces simultaneously, and adding and deleting

interfaces dynamically has many advantages mentioned in this chapter:

a) Transparent mobility: as the mobile host changes location, interfaces are added and deleted as they become available or fade.
b) Higher throughput: by adding the bandwidth of individual interfaces.
c) Informed choice: it may not be clear which interface will have the best end-to-end throughput. The best choice will vary depending on the rate of the interface, the number of users (if the interface is shared as in wireless Ethernet), the congestion of the end-to-end path, etc. By measuring the end-to-end throughput, the mobile will never choose the wrong interface.
d) Smooth hand-off: no blackout will occur if at least one interface remains connected while the set of available interfaces change
e) Connection diversity: the interference modes of different interfaces may not be correlated, which means that if an interface is exhibiting poor throughput due to jamming another interface may be immune to it (e.g. radio and infrared)

The experiments have shown that R-MTP is successful in aggregating bandwidth from different interfaces, by comparing the user-space implementation of R-MTP with the standard TCP implementation in the RedHat Linux distribution. Although under loss-less conditions the gains may not be expressive if the interfaces are of very dissimilar transmission rates, when losses are present even a low bandwidth interface may yield expressive gains, by creating a feed-back path that is not affected by the same loss incidents.

There are many aspects that merit further research. The first is to relax the assumption that bulk data is being transmitted. There are applications where a continuous flow of data is not always available, such as remote terminals (telnet, ssh). It is unclear if R-MTP would be advantageous for these applications, because the cost of maintaining multiple interfaces open may be too high for a low bandwidth application. The second is the development of an in-kernel R-MTP, to take advantage of the better timer granularity available for kernel modules when compared to user applications. This would allow higher speed interfaces to be used, and is in-line with the current practice of deploying both IEEE 802.11g and IEEE 802.11a in the same areas. Coupled with the development of the Link Layer Manager for notifying R-MTP of changes in the set of available interfaces, the in-kernel module allows for testing the mobility aspects of R-MTP.

# Chapter 7 Conclusion and Future Work

The main contribution of this Thesis is to create a new approach to host mobility, using the current IP infrastructure without changes, and adding mobility algorithms at the transport layer. This architecture breaks the strict layering scheme by making the transport layer *link layer-aware*. The coupling of knowledge of link resources with the path characteristics information available to transport protocols allows this mobility architecture to perform better than mobility solutions based on the network layer. At the same time, being on a different layer, this allows this architecture to coexist with mobility solutions implemented at the network and link layers. Optimal performance, though, should be achieved once the transport protocols have complete control over handoffs.

The complete mobility solution is composed of a framework for transport protocols and a location service. Combining all available link layers sub-channels creates a single virtual channel, which is the interface seen by the application. This *channel* abstraction creates a good framework for building transport protocols for mobile systems that share the characteristics of bandwidth aggregation, congestion avoidance and special mechanisms for dealing with losses on wireless links. The mechanisms used to this end are inverse multiplexing and traffic shaping (rate-based transmission). The experimental results obtained so far show that protocols created using the *channel* abstraction can be used for very different purposes, such as completely reliable communication for bulk data transmission or partially reliable, prioritized communication for multimedia with equally

satisfactory results.

Mobile nodes using wireless communication operate in less than ideal environments. Low bandwidth links limit application throughput, lossy channels challenge transport protocols optimized for congestion-based losses, and constantly changing channel availability causes breaks in transmission. The protocols presented in the last two Chapters address these problems using end-to-end channel aggregation. By exposing information about link-layer connectivity to the transport layer it is possible to use multiple channels concurrently, and adapt both intra-channel by varying the rate in which data is transmitted and extra-channel by adding and deleting channels from the pool of available channels. The resulting virtual channel has many characteristics that are more amenable to traffic, be it multimedia or best effort, than each of the channels taken alone. The virtual channel has more bandwidth, less overall variability and more resilience. The use of multiple channels can also alleviate the effect of losses on one particular channel. Finally, the use of multiple link layer technologies provides smooth handoffs as the use migrates through coverage areas of different wireless technologies.

The congestion control mechanism described in Chapter 3 is used by both R-MTP and MMTP. The Homeostatic Congestion Controller uses packet pair probing for bandwidth measurement, jitter correction for congestion avoidance and multiplicative rate decrease for congestion control. The term homeostatic comes from the two balancing forces, probing, which tends to overestimate bandwidth, and jitter correction, which pushes the rate back to the operating point. HCC was shown to be stable, fair and TCP friendly

through simulation.

In Chapter 4, a simple approach to loss discrimination based on accurate determination of congestion in the path from the sender to receiver was proposed. Congestion determination is based on simple observations of the arrival time of packets at the receiver. First, by monitoring the recent history of jitter, network loading can be monitored and congestion can be prevented by reducing the sending rate at times of increased load. Second, packet loss due to congestion causes subsequent packets to arrive early, causing "negative jitter," allowing such losses to be tagged as congestion losses. Lost packets that do not cause changes in the expected arrival times can be safely tagged as transmission losses. Transport protocols can then react wisely to loss, by reducing the sending rate in the presence of congestion, and maintaining the current rate in case of transmission loss.

The simulations show that the accuracy of the heuristic is directly tied to the regularity of the flow. Using a CBR stream, the heuristic was more accurate than with the time-varying flow. This observation may indicate that ack-clocked protocols such as TCP may not obtain good results using this heuristic because of their high variance in transmission times and their inherent burstiness. On the other hand, rate-based protocols may greatly benefit from the use of such heuristics to control their flow.

In Chapter 6, the experiments demonstrate the performance of the Reliable Multiplexing Transport Protocol (R-MTP) in various wireless environments. Despite its rate-based design, R-MTP fairly shares resources with TCP, which is ack-clocked, and also with

other R-MTP streams. The successful bandwidth aggregation across dissimilar links supports the intuition that adding bandwidth to the bottleneck link increases throughput for the application. Most importantly, it was demonstrated that bandwidth aggregation can be beneficial in some unintuitive scenarios. Specifically, it was shown that aggregation across two channels with an order of magnitude difference in capacity is beneficial if the high bandwidth channel is experiencing significant loss. This comes as a result of diversity. Using multiple interfaces not only results in link layer diversity, which may uncouple the occurrence of transmission losses, but also results in path diversity, which may uncouple the occurrence of congestion losses.

The simultaneous use of multiple interfaces brings many opportunities and challenges. The Multimedia Multiplexing Transport Protocol (MMTP) was designed for multimedia traffic, and reflects the constraints of this type of data stream. MMTP may perform better than other transport protocols by its adaptation to wireless environments (its congestion control being able to discern losses caused by the wireless transmission) but specially by being able to use multiple channels simultaneously. It is clear that a complementary communication framework would help the mobile hosts to take advantage of the benefits of such a protocol. The communication framework can help the mobile locate the available communication services. A path for future work lies in the integration of such a communication framework with the multiplexing protocols to be able to take advantage of knowledge of multiple link-layer channels.

With the groundwork established on the suite of transport protocols, the future work can

be centered on the larger architecture. This architecture encompasses the Link Layer Manager and the location service. After the design and implementation of these modules, testing the mobility characteristics of the protocols can begin. Another important issue is energy consumption. It may be possible to offset the higher power usage, caused by the concurrent use of multiple interfaces, with power savings by completing communication tasks faster. Another way to save energy is to create a larger infrastructure capable of mapping the availability of wireless communication in each area. This can prevent the mobile wasting energy by searching for an infrastructure that is absent.

# References


[AG000] A. Aggarwal, S. Savage, and T. Anderson. Understanding the performance of TCP pacing. In Proceedings of the IEEE INFOCOM, pages 1157--1165, Tel-Aviv, Israel, March 2000.

[AH099] B. Ahlgren, M. Bjorkman, and B. Melander. Network Probing Using Packet Trains. Submitted to Global Internet '99, March 1999.

[AL099] M. Allman, V. Paxson, W. Stevens, "TCP Congestion Control", Request for Comments: 2581, April 1999.

[AP004] J.G. Apostolopoulos, M. Trott , “Path Diversity for Enhanced Media Streaming”, IEEE Communication Magazine special issue on “Proxy Support for Streaming on the Internet”, 2004.

[BA095] H. Balakrishnan, S. Seshan, E. Amir, and R. Katz, "Improving TCP/IP Performance over Wireless Networks," in First ACM International Conference on Mobile Computing and Networking (MOBICOM), 1995.

[BA096] H. Balakrishnan, V. Padmanabhan, S. Seshan, and R. Katz, "A Comparison of Mechanisms for Improving TCP Performance over Wireless", in Proceedings of the SIGCOMM '96 Symposium, 1996.

[BA098] H. Balakrishnan and R.H. Katz, "Explicit Loss Notification and Wireless Web Performance," Proceedings of the IEEE Globecom Internet Mini-Conference, Sydney, Australia, November 1998.

[BA995] A. Bakre and B. R. Badrinath, "I-TCP: Indirect TCP for Mobile Hosts," in IEEE International Conference on Distributed Computing Systems (ICDCS) '95, 1995.

[BE004] Eric van den Berg, Sunil Madhani, Tao Zhang. "Real-Time Comparison of Quality of Interfaces (QoI) by Mobile Devices over Heterogeneous Radio Networks," First International Conference on Quality of Service in Heterogeneous Wired/Wireless Networks (QSHINE'04) pages. 206-215, 2004.

[BE104] Massimo Bernaschi, Filippo Cacace, Giulio Iannello. "Vertical Handoff Performance in Heterogeneous Networks," *icppw*, vol. 00, no. , pp. 100-107, 2004.

[BH001] Pravin Bhagwat, "Bluetooth: Technology for Short-Range Wireless Apps", IEEE Internet Computing, Vol. 5, No. 3, May-June 2001.

[BI098] S. Biaz and N. Vaidya, "Distinguishing congestion losses from wireless transmission losses: a negative result," in Proc. 7th Int. Conf. Computer Communications and Networks, Lafayette, LA, Oct. 1998, pp. 722-731.

[BI099] S. Biaz and N. H. Vaidya, "Discriminating Congestion Losses from Wireless Losses using Inter-Arrival Times at the Receiver," presented at IEEE Symposium ASSET'99, 1999.

[BL002] Blondia, C.; Van den Wijngaert, N; Willems, G; Casals, O.; "Performance analysis of optimized smooth handoff in mobile IP", Proceedings of the 5th ACM international workshop on Modeling analysis and simulation of wireless and mobile systems, 2002.

[BL098] Blake, S., Black, D., Carlson, M., Davies, E., Wang, Z. and W. Weiss, "An Architecture for Differentiated Services" RFC 2475, December 1998.

[BO099] Boerger E., “High Level System Design and Analysis using Abstract State Machines.”in: Hutter, D., Stephan, W., Traverso, P., Ullmann, M. (eds): Current Trends in Applied Formal Methods (FM-Trends 98). Lecture Notes in Computer Science, Vol. 1641, pp. 1-43. Springer-Verlag, Berlin Heidelberg New York (1999).

[BO093] Jean-Chrysostome Bolot. “End-to-end packet delay and loss behavior in the internet”, In SIGCOMM '93 Conference Proceedings, pages 289--298, San Francisco, California, USA, September 13--17, 1993.

[BO098] J. Bolot and T. Turletti, “Experience with Rate Control Mechanisms for Packet Video in the Internet.” ACM Computer Communications Review, vol. 28, pp. 4-15, 1998.

[BR095] L. S. Brakmo and L. L. Peterson, "TCP Vegas: End to end congestion avoidance on a global internet," IEEE Journal on Selected Areas in Communications, vol. 13, 1995.

[BR096] Brown K, Singh S, "M-TCP: TCP for mobile cellular networks" INFOCOM96, 1996.

[BR097] Braden, R., Zhang, L., Berson, S., Herzog, S. and S. Jamin, "Resource ReSerVation Protocol (RSVP) -- Version 1 Functional Specification", RFC 2205, September 1997.

[BR098] B. Braden, D. Clark, J. Crowcroft, B. Davie, S. Deering, D. Estrin,S. Floyd, V. Jacobson, G. Minshall, C. Partridge, L. Peterson, K. Ramakrishnan, S. Shenker, J. Wroclawski, L. Zhang, "Recommendations on Queue Management and Congestion Avoidance in the Internet", Request for Comments: 2309, April 1998.

[CA002] A. T. Campbell, J. Gomez, S. Kim, Z. Turanyi, A. G. Valko, C-Y. Wan, "Internet micromobility", Journal of High Speed Networks, IOS Press, (11) pp177-198, 2002.

[CA004] Dirceu Cavendish , Mario Gerla , Saverio Mascolo, “A control theoretical approach to congestion control in packet networks”, IEEE/ACM Transactions on Networking (TON), v.12 n.5, p.893-906, October 2004.

[CA095] R. Caceres and L. Iftode, "Improving the Performance of Reliable Transport Protocols in Mobile Computing Environments", IEEE Journal on Selected Areas in Communications, vol. 13, 1995.

[CA096] R. Carter and M. Crovella, "Measuring Bottleneck Link Speed in Packet-Switched Networks," Performance Evaluation, Vol. 27-8, pp. 297-318, Oct. 1996.

[CH098] F. M. Chiussi, D. A. Khotimsky, and S. Krishnan, “Generalized inverse multiplexing of switched ATM connections”, in Proceedings of the IEEE Conference on Global Communications (GlobeCom '98), 1998.

[CH002] H. Chen and Lj. Trajkovic, "Route optimization in mobile IP," Proc. IEEE Workshop on Wireless Local Networks, WLN 2002, Tampa, FL, Nov. 2002, pp. 847-848.

[CL088] D. D. Clark, M. Lambert, and L. Zhang, "NETBLT: A High Throughput Transport Protocol", RFC 998, March 1987.

[CL090] D. D. Clark and D. L. Tennenhouse, “Architectural Considerations for a New Generation of Protocols”, in Proceedings of the SIGCOMM '90 Symposium, 1990, pp. 200-208.

[CL000] William S. Cleveland and Don X. Sun, “Internet Traffic Data”, Journal of the American Statistical Association, 95, 979-985, 2000.

[CO095] D. E. Comer, “Internetworking with TCP/IP: Principles, Protocols and Architecture”, vol. 1: Prentice Hall, 1995.

[DA000] Jonathan Davidson, James Peters, Brian Gracely, “Voice over IP Fundamentals”, ISBN: 1578701686 , Cisco Press; 1st edition , March 27, 2000.

[DE098] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.

[DI003] Markus Dillinger, Kambiz Madani, Nancy Alonistioti, "Software Defined Radio Architectures, Systems and Functions (Wiley Series in Software Radio)", ISBN:0470851643, John Wiley & Sons, June 2003.

[DI080] Digital Equipment Corporation, Intel Corporation, Xerox Corporation,"The Ethernet - A Local Area Network", Version 1.0, September 1980.

[DO000] Dorgham Sisalem and Henning Schulzrinne, "The Direct Adjustment Algorithm: A TCP-Friendly Adaptation Scheme", in Quality of Future Internet Services, Berlin, Germany, September 2000.

[DO001] Dovrolis, C., P. Ramanathan, and D. Moore: 2001, "What do packet dispersion techniques measure?", In: IEEE Infocom'01. Anchorage, Alaska.

[DO099] A. Downey, "Using pathchar to estimate Internet link characteristics, Proc. SIGCOMM 1999, Cambridge, MA, pp. 241-250, Sept. 1999.

[DR097] R. Droms, "Dynamic Host Configuration Protocol", RFC 2131, March 1997.

[ER003] Ericsson AB, “EDGE Introduction of high-speed data in GSM/GPRS networks”, http://www.ericsson.com/products/white_papers_pdf/edge_wp_technical.pdf, 2003.

[FA096] K. Fall and S. Floyd, "Simulation-based Comparisons of Tahoe, Reno, and SACK TCP," Computer Communications Review, 1996.

[FA000] D. Farinacci, T. Li, S. Hanks, D. Meyer, P. Traina, “Generic Routing Encapsulation (GRE)”, Request for Comments: 2784, March 2000.

[FL000] Sally Floyd, Mark Handley, Jitendra Padhye, Joerg Widmer, "Equation-Based Congestion Control for Unicast Applications", ACM SIGCOMM 2000, Stockholm, Aug 2000.

[FL093] S. Floyd and V. Jacobson, "Random Early Detection Gateways for Congestion Avoidance," IEEE/ACM Transactions on Networking, 1993.

[FL094] S. Floyd, "TCP and Explicit Congestion Notification," ACM Computer Communications Review, vol. 24, 1994.

[FL100] Floyd, S., Mahdavi, J., Mathis, M. and M. Podolsky, "An Extension to the Selective Acknowledgment (SACK) Option for TCP", RFC 2883, July 2000.

[FO097] G. Folland and A. Sitaram, “The Uncertainty Principle: A Mathematical Survey”, Journal of Fourier Analysis and Applications, 1997 pp 207-238.

[FU093] V. Fuller, T. Li, J. Yu, K. Varadhan, "Classless Inter-Domain Routing (CIDR): an Address Assignment and Aggregation Strategy" - RFC 1519 September 1993.

[GE004] Mario Gerla, Bryan K. F. Ng, M. Y. Sanadidi, Massimo Valla, Ren Wang "TCP Westwood with adaptive bandwidth estimation to improve efficiency/friendliness tradeoffs ", Journal of computer communications, Volume 27, Number 1, January 2004.

[GO002] N. Gogate, D. Chung, S.S. Panwar, and Y. Wang, “Supporting image/video applications in a mobile multihop radio environment using route diversity and multiple description coding,” IEEE Trans. CSVT, pp.777–792, Sept 2002.

[GO102] L. Golubchick, J.C.S. Lui, T.F. Tung, A.L.H. Chow, W.-J. Lee, G. Franceschinis, and C. Anglano. Multi-path Continuous Media Streaming: What Are the Benefits? Performance Evaluation, 49(1-4), pp. 429-449, September 2002.

[GR004] Griffiths, David J., "Introduction to Quantum Mechanics", (2nd ed.), ISBN 013805326X, Prentice Hall, (2004).

[GU000] Yuri Gurevich, "Sequential Abstract State Machines Capture Sequential Algorithms", ACM Transactions on Computational Logic, vol. 1, no. 1, July 2000, 77-111.

[HA001] K. Harfoush, A. Bestavros, and J. Byers. “Measuring Bottleneck Bandwidth of Targeted Path Segments”, Technical Report BUCS-TR-2001-016, Boston University, Computer Science Department, July 2001.

[HA002] Hiroshi Harada, Ramjee Prasad, "Simulation and Software Radio for Mobile Communications", ISBN: 1580530443, Artech House Publishers, May 2002.

[HA003] M. Handley, S. Floyd, J. Padhye, J. Widmer, "TCP Friendly Rate Control (TFRC):Protocol Specification", RFC 3448, January 2003.

[HA005] Hanrahan H.E., “Integrated Digital Communications”, School of Electrical and Information Engineering, University of the Witwatersrand, Johannesburg, 2005.

[HA099] Handley, M., Schulzrinne, H., Schooler, E., and J. Rosenberg, "SIP: Session Initiation Protocol", RFC 2543, March 1999.

[HE027] W. Heisenberg, "Über den anschaulichen Inhalt der quantentheoretischen Kinematik und Mechanik", Zeitschrift für Physik, 43 1927, pp 172-198.

[HI093] R. Hinden, "Applicability Statement for the Implementation of Classless Inter-Domain Routing (CIDR)", Request for Comments: 1517, September 1993.

[HS003] Hsieh, R., Zhou, Z.-G., and Seneviratne, A. “S-MIP: A Seamless Handoff Architecture for Mobile IP”, In Proceedings of INFOCOM, San Francisco, USA, 2003.

[IE003] IEEE 802.11g-2003 IEEE Standard for Information technology - Telecommunications and information exchange between systems - Local and metropolitan area networks—Specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications - Amendment 4: Further Higher-Speed Physical Layer Extension in the 2.4 GHz Band.

[IE004] IEEE Standard 802.16-2004 IEEE Standard for Local and Metropolitan Area Networks - Part 16: Air Interface for Fixed Broadband Wireless Access Systems.

[IE099] IEEE 802.11, 1999 Edition (ISO/IEC 8802-11: 1999) IEEE Standards for Information Technology -- Telecommunications and Information Exchange between Systems -- Local and Metropolitan Area Network -- Specific Requirements -- Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications.

[IE199] IEEE 802.11a-1999 (8802-11:1999/Amd 1:2000(E)), IEEE Standard for Information technology - Telecommunications and information exchange between systems - Local and metropolitan area networks - Specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications - Amendment 1: High-speed Physical Layer in the 5 GHz band.

[IE299] IEEE 802.11b-1999 Supplement to 802.11-1999,Wireless LAN MAC and PHY specifications: Higher speed Physical Layer (PHY) extension in the 2.4 GHz band .

[JA084] R. Jain, D. Chiu, and W. Hawe, "A Quantitative Measure Of Fairness And Discrimination For Resource Allocation In Shared Computer Systems", DEC Research Report TR-301, September 1984.

[JA088] V. Jacobson, “Congestion avoidance and control”, In Proceedings of the SIGCOMM ’88 Symposium, pages 314-329, Aug. 1988.

[JE085] E. D. Jensen, C. D. Locke, and H. Tokuda, “A Time-Driven Scheduling Model for Real-Time Operating Systems”, in IEEE Real-Time Systems Symposium, 1985.

[JE099] K. Jeffay, “Towards a Better-Than-Best-Effort Forwarding Service for Multimedia Flows”. IEEE Multimedia, vol. 1999, 1999.

[KE000] Jun Ke and Carey Williamson, "Towards a Rate-Based TCP Protocol for the Web", Proceedings of the 8th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems, 2000.

[KE092] S. Keshav, "A Control-Theoretic Approach to Flow Control," presented at Proceedings of the SIGCOMM '92 Symposium, 1992.

[KL004] T.E. Klein, K. Leung, H. Zheng, "Enhanced Scheduling Algorithms for Improved TCP Performance in Wireless IP Networks", IEEE Globecom 2004.

[KO002] David Kotz and Kobby Essien, "Analysis of a Campus-wide Wireless Network", in Proceedings of the Eighth Annual International Conference on Mobile Computing and Networking (MobiCom), pages 107--118, September 2002.

[KR004] Krishnan, R., Sterbenz, J. P., Eddy, W. M., Partridge, C., and Allman, M., “Explicit transport error notification (ETEN) for error-prone wireless and satellite networks”, Comput. Networks 46, 3 (Oct. 2004), 343-362.

[LA000] Kevin Lai, Mary Baker, "Measuring Link Bandwidth Using a Deterministic Model of Packet Delay," SIGCOMM '2000, Aug 2000.

[LA001] Kevin Lai and Mary Baker. “Nettimer: A tool for Measuring Bottleneck Link Bandwidth”, In Proceedings of USENIX Symposium on Internet Technologies and Systems, March 2001.

[LA002] A. Lahanas and V. Tsaoussidis, "Additive Increase Multiplicative Decrease - Fast Convergence (AIMD-FC)", In Proccedings of Networks 2002 , Atlanta, Georgia, 26-29 August 2002.

[LE000] K.-W. Lee, R. Puri, T. Kim, K. Ramchandran, and V. Bharghavan, “An Integrated Source Coding and Congestion Control Framework for Video Streaming in the Internet”. Presented at IEEE Infocom 2000, 2000.

[LE005] Patrick P.C. Lee, Vishal Misra, and Dan Rubenstein. “Distributed Algorithms for Secure Multipath Routing”, Proceedings of IEEE INFOCOM, March 2005.

[LI004] X. Liu, K. Ravindran, B. Liu, and D. Loguinov, "Single-Hop Probing Asymptotics in Available Bandwidth Estimation: A Sample-Path Analysis, " ACM IMC, October 2004.

[LI093] T. Li, Y. Rekhter, "An Architecture for IP Address Allocation with CIDR" - RFC 1518 September 1993.

[LO003] Dmitri Loguinov , Hayder Radha, “End-to-end rate-based congestion control: convergence properties and scalability analysis”, IEEE/ACM Transactions on Networking (TON), v.11 n.4, p.564-577, August 2003.

[LU000] R. Ludwig and R. H. Katz, "The Eifel Algorithm: Making TCP Robust Against Spurious Retransmissions," ACM Computer Communications Review, vol. 30, 2000.

[MA001] L. Magalhaes and R. Kravets, “MMTP: Multimedia Multiplexing Transport Protocol”. in The First Workshop on Data Communications in Latin America and the Caribbean (SIGCOMM-LA 2001), 2001, San Jose, Costa Rica. Supplement to Computer Communication Review. New York : The Association for Computer Machinery, 2001. v. 31. p. 220-243.

[MA003] Magalhaes, Luiz ; Kravets, Robin ; Harris, Albert, “Improving Performance of Rate-Based Transport Protocols in Wireless Environments”, In: XX Simposio Brasileiro de Telecomunicacoes, 2003, Rio de Janeiro. Anais do XX Simposio Brasileiro de Telecomunicacoes, 2003.

[MA005] El Malki, "Low Latency Handoffs in Mobile IPv4", INTERNET-DRAFT, July 2005

[MA096] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow, "TCP Selective Acknowledgement Options," Internet Engineering Task Force, Request for Comments RFC 2018, October 1996 1996.

[MA097] Mathis, Semke, Mahdavi, Ott, "The macroscopic behavior of the TCP congestion avoidance algorithm", in Computer Communication Review, 27(3), July 1997.

[MA099] L. Mamakos, K. Lidl, J. Evarts, D. Carrel, D. Simone, R. Wheeler, "A Method for Transmitting PPP Over Ethernet (PPPoE)", RFC 2516 , February 1999.

[MA101] Magalhaes, Luiz ; Kravets, Robin, “End-to-End Inverse Multiplexing for Mobile Hosts”, Journal of the Brazilian Computer Society, Rio de Janeiro, v. 7, n. 2, p. 52-62, 2001.

[MA201] Magalhaes, Luiz ; Kravets, Robin, “Transport Level Mechanisms for Bandwidth Aggregation on Mobile Hosts”. In: The 9th International Conference on Network Protocols (ICNP 2001), 2001, Riverside, California. Preceedings of The 9th International Conference on Network Protocols. Los Alamitos, CA : IEEE Computer Society, 2001.

[MO003] Marek Malowidzki, "Simulation-based Study of ECN Performance in RED Networks", SPECTS'03, 2003.

[MI000] Joseph Mitola, "Software Radio Architecture", ISBN: 047138492-5, Wiley, October 2000.

[MO087] P. Mockapetris, "Domain Names--Implementation and Specification", RFC 1035 (Standard: STD 13) Nov-1987

[MO103] Marek Malowidzki, "ECN is Fine - But Will It Be Used?", PDCN'03, 2003

[MO187] P. Mockapetris, "Domain Names--Concepts and Facilities", RFC 1034 (Standard: STD 13) Nov-1987

[NA084] John Nagle, "Congestion Control in IP/TCP Internetworks", RFC 896, 6 January 1984.

[NA095] Klara Nahrstedt, “End-to-end QoS guarantees in networked multimedia systems”, ACM Computing Surveys (CSUR), v.27 n.4, p.613-616, Dec. 1995

[NA099] Klara Nahrstedt, “Quality of Service Guarantees in Networked Multimedia Systems”, Handbook on Multimedia Computing, Borko Fuhrt (Ed), pp. 839-875, CRC Press, 1999.

[NI000] Nissanka B. Priyantha, Anit Chakraborty, Hari Balakrishnan, "The Cricket Location-Support system", Proc. 6th ACM MOBICOM, Boston, MA, August 2000.

[OB001] Vincent Oberle, Uwe Walter, “Micro-second precision timer support for the Linux kernel”, Technischer Bericht, Nov 2001

[OM001] UML specification (version 1.4.2, OMG document: formal/05-04-01) ISO/IEC 19501.

[PA097] V. Paxson, "End-to-End Internet Packet Dynamics. In Proc. of ACM SIGCOMM, September 1997.

[PA099] J. Padhye, J. Kurose, D. Towsley, and R. Koodli, "A model based TCP-friendly rate control protocol," in Proc. International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV), Basking Ridge, NJ, June 1999.

[PA999] C. Parsa and J. J. Garcia-Luna-Aceves, "Improving TCP Congestion Control Over Internets with Heterogeneous Transmission Media," presented at IEEE International Conference on Network Protocols (ICNP'99), 1999.

[PE000] K. Pentikousis, "TCP in Wired-Cum-Wireless Environments," IEEE Communications Surveys, 2000.

[PE002] Perkins, C., "IP Mobility Support for IPv4", RFC 3220, January 2002.

[PE096] Perkins, C.,: “IP Encapsulation within IP”, RFC 2003, October 1996.

[PE098] C. Perkins, “Mobile IP: Design Principles and Practice”, ISBN: 0201634694, Addison-Wesley Longman, Reading, MA, 1998

[PL082] David C. Plummer, "An Ethernet Address Resolution Protocol", RFC 826, November 1982

[PO081] J. Postel, "Transmission Control Protocol", Internet Engineering Task Force, Request for Comments (Standard) RFC 793, September 1981.

[PO085] J. Postel, J. Reynolds, "File Transfer Protocol (FTP)", RFC 959, October 1985

[PU004] Puneet Sharma, Sung-Ju Lee, Jack Brassil, and Kang G. Shin, "Distributed Channel Monitoring for Wireless Bandwidth Aggregation" Proceedings of IFIP-TC6 Networking Conference 2004, Athens, Greece, May 2004, pp. 345-356.

[RA001] K. Ramakrishnan, S. Floyd, D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP", Request for Comments: 3168, September 2001

[RE002] Jeffrey H. Reed, "Software Radio: A Modern Approach to Radio Engineering", ISBN: 0130811580, Prentice Hall, May 2002.

[RE003] Pierre Reinbold and Olivier Bonaventure; "IP micro-mobility protocols"; IEEE Communications Surveys and Tutorials, 2003

[RE099] Reza Rejaie, Mark Handely and Deborah Estrin, "RAP: An End-to-end Rate-based Congestion Control Mechanism for Realtime Streams in the Internet", Proc. IEEE Infocom'99, 1999

[RI005] Bruno Ribeiro, Edmundo de Souza e Silva, Don Towsley, "On the Efficiency of Path Diversity for Continuous Media Applications" UMass CMPSCI Technical Report 05-19. 2005.

[RO000] J. Rosenberg, H. Schulzrinne, “A Framework for Telephony Routing over IP”, RFC 2871, June 2000.

[RO090] Rose, M. T., "The Open Book: A Practical Perspective on OSI", ISBN: 0136430163, Englewood Cliffs, NJ: Prentice-Hall. (1990).

[SA084] J. H. Saltzer, D. P. Reed, and D. D. Clark. “End-to-end arguments in system design”, ACM Transactions on Computer Systems, pages 277288, 1984

[SC096] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: a transport protocol for real-time applications", Request for Comments (Proposed Standard) 1889, Internet Engineering Task Force, Jan. 1996.

[SE099] S. Servetto and K. Nahrstedt, “Broadcast Quality Video over IP”, IEEE Journal on Selected Areas in Communications, Special issue on QoS in the Internet, 1999.

[SI093] A. DeSimone, M. C. Chuah, and O. C. Yue, "Throughput Performance of Transport-Layer Protocols over Wireless LANs," presented at Globecom '93, 1993.

[SI094] W. Simpson, "The Point-to-Point Protocol (PPP)", Request for Comments: 1661 (STD: 51), July 1994.

[SI099] P. Sinha, N. Venkitaraman, R. Sivakumar, and V. Bharghavan, “WTCP: A Reliable Transport Protocol for Wireless Wide-Area Networks”, in ACM Mobicom '99, 1999.

[SK096] K. Sklower, B. Lloyd, G. McGregor, D. Carr, T. Coradetti, "The PPP Multilink Protocol (MP)", Request for Comments: 1990, August 1996

[SO097] James D. Solomon, “Mobile IP: the Internet unplugged”, ISBN: 0138562466, Prentice-Hall, Inc., Upper Saddle River, NJ, 1997

[SN000] A. C. Snoeren, H. Balakrishnan, "An End-to-End Approach to Host Mobility", Proc. ACM MobiCom'00, pp. 155 – 166, Boston, USA, Aug 2000.

[SN001] A. Snoeren, H. Balakrishnan and M.F.Kaashoek , "The Migrate Approach to Internet Mobility", Proc. Of the Oxygen Student Workshop, July 2001.

[SN101] Alex C. Snoeren, Hari Balakrishnan, and M. Frans Kaashoek, "Reconsidering IP Mobility", Proc. 8th HotOS, May 2001.

[ST000] R. Stewart, Q. Xie, K. Morneault, C. Sharp H. Schwarzbauer, T. Taylor, I. Rytina,M. Kalla, L. Zhang, V. Paxson, "Stream Control Transmission Protocol", Request for Comments: 2960, October 2000

[ST100] Stallings, W., “Data and Computer Communications”, Prentice-Hall, New Jersey, Sixth Edition, 2000.

[ST002] William Stallings, "Wireless Communications and Networks", ISBN: 0-13-040864-6, Englewood Cliffs, NJ:Prentice Hall (2002)

[ST098] M. Stemm and R. H. Katz, “Vertical Handoffs in Wireless Overlay Networks”, ACM Mobile Networking (MONET), Special Issue on Mobile Networking in the Internet, 1998.

[ST198] Stangel M, Bhargavan V "Improving TCP performance in mobile computing environments" Proc. IEEE ICC'98, 1998

[TA003] Andrew Tanenbaum, "Computer Networks", ISBN: 0-13-066102-3, Englewood Cliffs, NJ:Prentice Hall (2003)

[TE005] Douglas Vidal Teixeira, Luiz Cláudio Schara Magalhaes, "A Free Software Streaming Video Application based on MMTP", FISL 6.0 - 6th International Forum on Free Software, Porto Alegre, Brasil, June 2005.

[TO002] Jean Tourrilhes, Luiz Magalhaes, Casey Carter, “On-Demand TCP: Transparent peer to peer TCP/IP over IrDA”. In: IEEE International Conference on Communication, 2002, New York. Proceedings of ICC 2002. Los Alamitos, CA : IEEE Computer Society, 2002.

[TO003] Jean Tourrilhes and Venky Krishnan, “Co-Link: Using wireless diversity for more than just connectivity”, Proc. of WCNC 2003.

[TU004] Walter H.W. Tuttlebee, "Software Defined Radio: Baseband Technologies for 3G Handsets and Basestations", John Wiley & Sons, February 2004.

[VA099] Vaidya N, Mehta M "Delayed duplicate acknowledgements: A TCP-unaware approach to improve performance of TCP over wireless" Texas A&M University, Technical Report 99-003, February 1999

[VI095] Andrew J. Viterbi.. “CDMA: Principles of Spread Spectrum Communication”, (1st edition), ISBN 0201633744, Prentice Hall, 1995.

[VU001] S. Vutukury and J. J. Garcia-Luna-Aceves. “MDVA: A Distance-Vector Multipath Routing Protocol”, In Proceedings of the IEEE INFOCOM, pages 557--564, 2001.

[WA004] Bing Wang, Jim Kurose, Prashant Shenoy, Don Towsley, "Multimedia Streaming via TCP: An Analytic Performance Study", Proceedings of ACM Multimedia, New York City, October, 2004.

[WA005] Ren Wang, Kenshin Yamada, M. Yahya Sanadidi, and Mario Gerla " TCP with sender-side intelligence to handle dynamic, large, leaky pipes ", IEEE Journal on Selected Areas in Communications, 23(2):235-248, 2005.

[WE099] E. Wedlund and H. Schulzrinne, "Mobility Support using SIP", Proc. ACM WoWMoM'99, Seattle, USA, Aug 1999, pp. 76 - 82.

[XI002] W. Xing, H. Karl, and A. Wolisz, "M-SCTP: Design and prototypical implementation of an end-to-end mobility concept," 5th Intl. Workshop The Internet Challenge: Technology and Applications, Berlin, Germany, October 2002.

[XU098] Dongyan Xu, Baochun Li, Klara Nahrstedt, Jane W. S. Liu, Providing Seamless QoS for Multimedia Multicast in Wireless Packet Networks, in Proc. of SPIE International Symposium on Voice, Video, and Data Communications, Boston, MA, November, 1998

[XY001] Xylomenos, G., Polyzos, C. G., “TCP Performance Issues over Wireless Links”, IEEE Communications Magazine, April 2001.

[YA004] Kenshin Yamada, Ren Wang, and M.Y. Sanadidi and Mario Gerla " TCP Westwood with agile probing: Dealing with dynamic, large, leaky pipes ", In Processing of IEEE ICC. volume 2. pages 1070-1074. 2004

[ZA098] X. Zhao, C. Castelluccia, and M. Baker, “Flexible Network Support for Mobility”, in Fourth ACM International Conference on Mobile Computing and Networking (MOBICOM'98), 1998.

[ZH003] Shelley Zhuang, Kevin Lai, Ion Stoica, Randy Katz, and Scott Shenker. “Host Mobility Using an Internet Indirection Infrastructure”, In Proceedings of the First International Conference on Mobile Systems, Applications, and Services, pages 129--144, San Francisco, CA, May 2003. USENIX Association.

[ZH004] Yueping Zhang , Seong-Ryong Kang , Dmitri Loguinov, “Delayed stability and performance of distributed congestion control”, ACM SIGCOMM Computer Communication Review, v.34 n.4, October 2004.

# Author's Biography

Luiz Claudio Schara Magalhães was born in Rio de Janeiro, Brazil, on October 30, 1965. He is married to Isabela Melo Magalhães. He has a son Luiz Elias born on July 25, 2001. He graduated from Pontifícia Universidade Católica do Rio de Janeiro in 1989 with a degree in Electrical Engineering. He worked at Jornal do Brasil from 1988 to 1990, when he started his own software development company, Schara Sistemas. He completed a Master of Science in Computer Science from Pontifícia Universidade Católica do Rio de Janeiro in 1994. He worked as a consultant for the Brazilian Academic Research Network (RNP) until he passed in first place in the public admission process for Universidade Federal Fluminense, where he is a Professor at the Department of Telecommunications Engineering. In 1996 he was granted a leave of absence from his University and a grant from CNPq, and moved to Urbana, Illinois, to pursue graduate study in Computer Science, in the area of Computer Networks. In the University of Illinois at Urbana Champaign he was a Teaching Assistant, a Research Assistant and an Instructor. In 2001 his leave of absence ended, and he returned to UFF, where he teaches the subjects of network programming, operating systems, computer networks and network applications to undergraduate students, and computer networks and mobile computing to graduate students.

He is currently head of the area of Data Communications in the Department of Telecommunications Engineering, part of the Graduate Committee and of the Undergraduate Collegiate, runs the Telecommunications Laboratory and the MediaCom

Laboratory, and is a consultant in computer networks for the Núcleo de Tecnologia de Informação.

His current research interests are varied. He has received two grants from the Brazilian Government, one to study media streaming for the Brazilian Digital TV standard development, and another to study transport protocols for high speed networks. He and his students are also doing research on a new model for electronic mail, movement patterns on cellular networks with the aid of a Brazilian cellular carrier, medical imaging data management and collaborative environments for telemedicine with the aid of the staff of the Hospital Universitário Antônio Pedro, frequency reuse on mesh networks and metropolitan deployment of voice over IP.